FlowPINNs: A Variational Framework for PDE Parameter Inference and Uncertainty Quantification

David Dalton **Hao Gao** **Dirk Husmeier**
University of Glasgow

Abstract

Inverse problems for parameter identification in systems governed by partial differential equations (PDEs) arise in many areas of science and engineering. Traditionally, such problems have been addressed using classical numerical methods. More recently, physics-informed neural networks (PINNs) have emerged as a promising alternative for learning PDE-constrained models directly from data. However, providing principled uncertainty quantification (UQ) for the predictions obtained using PINNs remains a significant challenge. To address this limitation, we introduce flowPINNs, a probabilistic framework for estimation and UQ in PDE parameter inverse problems. The central idea is to define a variational posterior that combines a normalising flow approximation for the distribution over the PDE parameters with a parameterised PINN representing the corresponding PDE solution. This joint formulation enables efficient posterior inference via maximisation of the evidence lower bound (ELBO), thereby casting the inverse problem as a tractable optimisation task. Through a series of numerical experiments, we demonstrate that flowPINNs can achieve improved performance with strong computational efficiency when compared to existing UQ approaches for PINNs.

1 INTRODUCTION

Parameter inverse problems involving partial differential equations (PDEs) aim to identify latent physical parameters from indirect, noisy, or incomplete observations (Kaipio and Somersalo, 2005), and they arise in many scientific and engineering applications (Arridge et al., 2019; Stuart, 2010; Gao et al., 2015). Such problems have traditionally been addressed by embedding a forward PDE solver, such as the finite-element method (FEM) (Sabat and Kundu, 2020), within a parameter calibration procedure that aligns model predictions with observational data. However, this approach can be computationally expensive, as accurate parameter estimation typically requires hundreds or thousands of repeated evaluations of the numerical solver (Karniadakis et al., 2021a).

Physics-informed machine learning (PIML) has emerged in recent years as an alternative paradigm for PDE modelling (Cuomo et al., 2022). PIML integrates knowledge of the underlying PDE directly into a machine learning model, which can yield improved interpretability, accuracy, and generalisability over purely data-driven approaches (Rackauckas et al., 2020; Ryan et al., 2026). Among PIML methods, physics-informed neural networks (PINNs) (Raissi et al., 2019) have become the most widely adopted framework. A PINN represents the physical field of interest with a neural network surrogate whose objective function incorporates the governing PDE residual, thereby enforcing physical consistency while avoiding the need for an explicit numerical solver.

Uncertainty quantification (UQ) in PDE inverse problems is critical for assessing the credibility of predictions, guiding data acquisition, and supporting robust decision-making under uncertainty (Peherstorfer et al., 2018; Psaros et al., 2023). Several probabilistic PIML approaches have been proposed to address this challenge, most notably Bayesian PINNs (B-PINNs) (Yang et al., 2021). By embedding PDE knowledge into a probabilistic learning process, B-PINNs can provide more informative and reliable solutions than deterministic alternatives. Nevertheless, inference in B-PINNs remains challenging. Markov chain Monte Carlo (MCMC) methods are asymptotically exact but computationally expensive, and typically sensitive to measurement noise

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

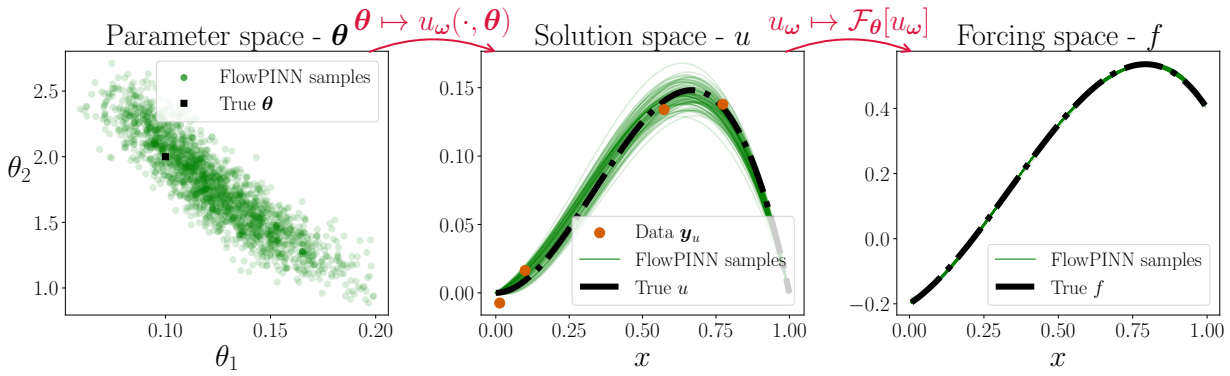


Figure 1: Samples from a trained flowPINN (see Eq. 10) for a one-dimensional reaction diffusion PDE, $\mathcal{F}_{\boldsymbol{\theta}}$. A normalising flow $q_{\boldsymbol{\phi}}$ defines an approximate posterior distribution over the PDE parameters $\boldsymbol{\theta} = (\theta_1, \theta_2)$, from which 1000 samples have been drawn (left panel). Each sample $\boldsymbol{\theta}_i \sim q_{\boldsymbol{\phi}}$ is mapped to an associated PINN solution $u_{\boldsymbol{\omega}}(\cdot, \boldsymbol{\theta}_i)$ (centre panel), and forcing term $\mathcal{F}_{\boldsymbol{\theta}_i}[u_{\boldsymbol{\omega}}](\cdot, \boldsymbol{\theta}_i)$ (right panel). The uncertainty in $\boldsymbol{\theta}$ and u arises from noise in the sparse observation data. The comparative lack of uncertainty in f reflects the (soft) enforcement of the governing PDE by the flowPINN, whereby it is trained such that $\mathcal{F}_{\boldsymbol{\theta}_i}[u_{\boldsymbol{\omega}}](\cdot, \boldsymbol{\theta}_i) \simeq f(\cdot)$ for all $\boldsymbol{\theta}_i \sim q_{\boldsymbol{\phi}}$. Further discussion is provided in Appendix A.

hyperparameters. Variational inference (VI) offers a more efficient alternative, yet traditional factorised VI methods generally fail to achieve competitive performance in this setting.

To address these challenges, we introduce flowPINNs, a novel VI framework for inference and UQ in inverse problems involving PDE control parameters. A flowPINN combines a normalising flow with a PINN to define a joint variational posterior over the PDE parameters and solution function which mirrors the posterior structure obtained when using a traditional numerical solver. The model is trained against the evidence lower bound (ELBO) with respect to both observation data and PDE residual points. Through numerical experiments, we demonstrate that flowPINNs consistently outperform existing VI approaches and achieve performance comparable to, or exceeding, MCMC sampling, while offering improved computational efficiency. Moreover, flowPINNs infer noise hyperparameters adaptively, eliminating the manual tuning commonly required by B-PINNs.

2 BACKGROUND

2.1 Partial differential equations

Partial differential equations (PDEs) are mathematical models widely used in science and engineering to describe how physical quantities evolve over space and time. A scalar PDE can be represented in the general form

$$\mathcal{F}_{\boldsymbol{\theta}}[u](\boldsymbol{x}, t) = f(\boldsymbol{x}, t), \quad (\boldsymbol{x}, t) \in \Omega \times (0, T], \qquad (1)$$

where $\Omega \subset \mathbb{R}^D$ is the spatial domain, t denotes time, f is the forcing term (equal to zero for unforced systems), $\mathcal{F}_{\boldsymbol{\theta}}$ is a PDE operator with control parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subset \mathbb{R}^{N_p}$, and u is the solution function. Imposing additional constraints on u at the initial time $t = 0$ and along the domain boundary $\partial\Omega$ defines an initial-boundary value problem (IBVP). For well-posed PDEs, these constraints guarantee the existence and uniqueness of the solution to Eq. (1) (Strauss, 2007, Ch. 1.5).

The most fundamental task associated with an IBVP is the *forward* problem of computing the solution u, while *inverse* problems involve the inference of system properties from observational data. In this work, we focus on parameter inverse problems, where the goal is to infer both the PDE parameters $\boldsymbol{\theta}$ and solution u from a dataset of noisy measurements of u.

2.2 Physics-informed neural networks

A physics-informed neural network (PINN) (Raissi et al., 2019) is a deep learning model designed to model systems governed by PDEs. To solve a parameter inverse problem with a PINN, the solution function is represented by a neural network $u_{\boldsymbol{\omega}}(\boldsymbol{x}, t)$ with parameters $\boldsymbol{\omega}$. The PDE parameters $\boldsymbol{\theta}$ are jointly estimated with the network parameters $\boldsymbol{\omega}$ by minimising a multi-task loss function which typically comprises a data mismatch term, a PDE residual term, and a boundary/initial condition mismatch term:

$$L(\boldsymbol{\omega}, \boldsymbol{\theta}) = L_{\text{data}}(\boldsymbol{\omega}) + L_{\text{PDE}}(\boldsymbol{\theta}, \boldsymbol{\omega}) + L_{\text{b/i}}(\boldsymbol{\omega}), \qquad (2)$$

The PDE residual L_{PDE} measures the discrepancy between the true forcing f and the network-induced forc-

ing $\mathcal{F}_{\boldsymbol{\theta}}[u_{\boldsymbol{\omega}}]$ at a discrete set of collocation points in the spatio-temporal domain, with derivatives computed via automatic differentiation.

PINNs have been successfully applied across a wide range of scientific domains, including fluid dynamics (Jin et al., 2021), solid mechanics (Haghighat et al., 2021), and electromagnetics (Karniadakis et al., 2021b). In their standard form, however, PINNs do not provide UQ for either the inferred PDE parameters or the predicted solution. This limitation has spurred the development of several different UQ approaches for PINNs, which we review in Section 4.2.

2.3 Normalising Flows

Normalising flows construct flexible probability distributions by applying a sequence of invertible transformations to a simple base distribution (Prince, 2023, Ch. 16). Formally, let $\boldsymbol{z}_0$ be a random vector with a known density q_0 (typically Gaussian), and let $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_K$ be smooth, invertible functions. A normalising flow defines a distribution over the final transformed variable $\boldsymbol{z}_K$ through the recursion

$$\boldsymbol{z}_k = \boldsymbol{g}_k(\boldsymbol{z}_{k-1}), \quad k = 1, \ldots, K. \tag{3}$$

The invertibility of each $\boldsymbol{g}_k$ allows the log-density of $\boldsymbol{z}_K$ to be computed using the change-of-variables formula (Devore et al., 2012, Sec. 4.7):

$$\log q(\boldsymbol{z}_K) = \log q_0(\boldsymbol{z}_0) - \sum_{k=1}^{K} \log \left| \det \frac{\partial \boldsymbol{g}_k}{\partial \boldsymbol{z}_{k-1}} \right|. \tag{4}$$

Inverse Autoregressive Flows. Different flow architectures correspond to different parameterisations of the transformations $\boldsymbol{g}_k$ in Eq. (3). In this work, we use inverse autoregressive flows (IAFs) (Kingma et al., 2016), which are particularly effective for variational inference due to their efficient sampling properties (Papamakarios et al., 2017). In an IAF, each transformation $\boldsymbol{g}_k$ is defined using a masked autoencoder for distribution estimation (MADE) (Germain et al., 2015). MADE enforces the following autoregressive structure on the components of $\boldsymbol{z}_k$:

$$z_k^{(i)} = z_{k-1}^{(i)} \exp(\alpha_i) + \mu_i, \tag{5}$$

where the parameters are computed as

$$\mu_i = g_{\mu_i}\left(\boldsymbol{z}_{k-1}^{(1:i-1)}\right), \qquad \alpha_i = g_{\alpha_i}\left(\boldsymbol{z}_{k-1}^{(1:i-1)}\right), \tag{6}$$

with g_{μ_i} and g_{α_i} being neural networks that depend only on the preceding components $\boldsymbol{z}_{k-1}^{(1:i-1)} = (z_{k-1}^{(1)}, \ldots, z_{k-1}^{(i-1)})$. This autoregression ensures that the Jacobian of each transformation $\boldsymbol{g}_k$ is lower triangular, which allows the log determinant terms in Eq. (4) to be computed efficiently.

3 PROBLEM STATEMENT

We consider parameter inverse problems in which, given a PDE operator $\mathcal{F}_{\boldsymbol{\theta}}$ and specified initial and boundary conditions, the objective is to infer and provide Bayesian UQ for the control parameters $\boldsymbol{\theta}$ and solution function u. We assume that a dataset $D_u = \{([\boldsymbol{x}_u^{(i)}, t_u^{(i)}], y_u^{(i)})\}_{i=1}^{N_u}$ is available, where the observations $\boldsymbol{y}_u$ are corrupted by iid zero-mean Gaussian measurement noise:

$$y_u^{(i)} = u(\boldsymbol{x}_u^{(i)}, t_u^{(i)}) + \varepsilon_u^{(i)}, \;\; \varepsilon_u^{(i)} \overset{iid}{\sim} \mathcal{N}(0, \sigma_u^2). \tag{7}$$

We now review two existing approaches for solving this class of problem, which will serve as baselines for the method introduced in Section 4.

3.1 Conventional Numerical Solvers

The gold standard approach to solving PDE parameter inverse problems relies on a conventional numerical solver (CNS), such as the finite-element (Zienkiewicz et al., 2005), finite-difference (Smith, 1985), or finite-volume (Eymard et al., 2000) methods. A CNS can be used to define a likelihood function $\mathcal{L}_{\sigma_u}$ for the unknown parameters $\boldsymbol{\theta}$ (see Figure 2 a), which is evaluated in two stages. Firstly, for a given parameter value $\boldsymbol{\theta}$, the CNS computes the solution $u_{\boldsymbol{\theta}}$ that satisfies the IBVP. Secondly, predictions from $u_{\boldsymbol{\theta}}$ are passed through the Gaussian observation model to evaluate the likelihood. Combining this likelihood with a prior distribution $p(\boldsymbol{\theta})$ yields the posterior $p(\boldsymbol{\theta} \mid D_u) \propto \mathcal{L}_{\sigma_u}(\boldsymbol{\theta}; D_u)\, p(\boldsymbol{\theta})$. This posterior can be explored using a Markov chain Monte Carlo (MCMC) sampling algorithm (Gamerman, 2006), which provides full UQ over the parameters $\boldsymbol{\theta}$, and consequently over the solution function u.

A key drawback of this approach is that each evaluation of the likelihood $\mathcal{L}_{\sigma_u}$ requires solving the IBVP explicitly, which can be computationally expensive for systems governed by complex PDEs. This computational burden has motivated the development in recent years of PIML methods for inverse problems (Karniadakis et al., 2021a). PIML approaches avoid the need to repeatedly solve the governing PDE, potentially offering substantial computational gains (e.g. Buoso et al. (2021)).

3.2 Bayesian PINNs

The Bayesian PINN (B-PINN) (Yang et al., 2021) is a PIML method that can be used for PDE parameter inference and UQ. In this framework, the unknown solution of the PDE is represented by a neural network $u_{\boldsymbol{\omega}} \colon \bar{\Omega} \times [0, T] \to \mathbb{R}$, which serves as a flexible function approximator. The governing PDE is incorporated through a collocation dataset

a CNS: $\mathcal{L}_{\sigma_u}(\boldsymbol{\theta}; D_u, \sigma_u)$

Require: parameters $\boldsymbol{\theta}$; data D_u; hyperparameter σ_u.

$u_{\boldsymbol{\theta}} \leftarrow$ `SOLVE-IBVP`$(\boldsymbol{\theta})$
$\hat{\boldsymbol{y}}_u \leftarrow u_{\boldsymbol{\theta}}(\mathbf{X}_u)$
$\mathcal{L}_{data} \leftarrow \mathcal{N}(\hat{\boldsymbol{y}}_u, \sigma_u^2 \mathbf{I})[\boldsymbol{y}_u]$
Return $\mathcal{L}_{data}$

b B-PINN: $\mathcal{L}_{\boldsymbol{\sigma}}(\boldsymbol{\theta}, \boldsymbol{\omega}; D_u, D_f)$

Require: parameters $\boldsymbol{\theta}$, $\boldsymbol{\omega}$; data D_u, D_f; hyperparams σ_u, σ_f.

$\hat{\boldsymbol{y}}_u \leftarrow u_{\boldsymbol{\omega}}(\mathbf{X}_u)$
$\hat{\boldsymbol{y}}_f \leftarrow \mathcal{F}_{\boldsymbol{\theta}}[u_{\boldsymbol{\omega}}](\mathbf{X}_f)$
$\mathcal{L}_{data} \leftarrow \mathcal{N}(\hat{\boldsymbol{y}}_u, \sigma_u^2 \mathbf{I})[\boldsymbol{y}_u]$
$\mathcal{L}_{pde} \leftarrow \mathcal{N}(\hat{\boldsymbol{y}}_f, \sigma_f^2 \mathbf{I})[\boldsymbol{y}_f]$
return $\mathcal{L}_{data} \cdot \mathcal{L}_{pde}$

c flowPINN: $\mathcal{L}_{\boldsymbol{\omega},\boldsymbol{\sigma}}(\boldsymbol{\theta}; D_u, D_f)$

Require: parameters $\boldsymbol{\theta}$; data D_u, D_f; hyperparams $\boldsymbol{\omega}$, σ_u, σ_f.

$\hat{\boldsymbol{y}}_u \leftarrow u_{\boldsymbol{\omega}}(\mathbf{X}_u, \boldsymbol{\theta})$
$\hat{\boldsymbol{y}}_f \leftarrow \mathcal{F}_{\boldsymbol{\theta}}[u_{\boldsymbol{\omega}}](\mathbf{X}_f, \boldsymbol{\theta})$
$\mathcal{L}_{data} \leftarrow \mathcal{N}(\hat{\boldsymbol{y}}_u, \sigma_u^2 \mathbf{I})[\boldsymbol{y}_u]$
$\mathcal{L}_{pde} \leftarrow \mathcal{N}(\hat{\boldsymbol{y}}_f, \sigma_f^2 \mathbf{I})[\boldsymbol{y}_f]$
return $\mathcal{L}_{data} \cdot \mathcal{L}_{pde}$

Figure 2: Pseudo-code for evaluating the likelihood functions for a CNS, B-PINN and flowPINN. Inputs from D_u are labelled $\mathbf{X}_u$, and inputs from D_f are labelled $\mathbf{X}_f$. Each evaluation of the CNS likelihood $\mathcal{L}_{\sigma_u}$ requires a numerical solution to the IBVP. The B-PINN likelihood $\mathcal{L}_{\boldsymbol{\sigma}}$ avoids this, but greatly increases the dimensionality of the likelihood space by introducing the PINN parameters $\boldsymbol{\omega}$. A flowPINN combines aspects of both: its likelihood $\mathcal{L}_{\boldsymbol{\omega},\boldsymbol{\sigma}}$ is only defined over the PDE parameters $\boldsymbol{\theta}$, and the IBVP is never explicitly solved. This is achieved by treating $\boldsymbol{\omega}$ as a *hyperparameter* of $\mathcal{L}_{\boldsymbol{\omega},\boldsymbol{\sigma}}$, rather than a direct input.

$D_f = \{([\boldsymbol{x}_f^{(i)}, t_f^{(i)}], y_f^{(i)})\}_{i=1}^{N_f}$, where the modeller is free to select the locations of the input points to ensure good coverage of the spatio-temporal domain.

While observations of the solution typically contain noise, the collocation points are inherently noise-free because the PDE is assumed to be known exactly. Nevertheless, to define a well-behaved likelihood for Bayesian inference it is common to assume a small Gaussian observation noise,

$$y_f^{(i)} = f(\boldsymbol{x}_f^{(i)}, t_f^{(i)}) + \varepsilon_f^{(i)}, \quad \varepsilon_f^{(i)} \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_f^2), \tag{8}$$

where σ_f is a nugget hyperparameter introduced for numerical stability.

This formulation enables the construction of a joint likelihood $\mathcal{L}_{\boldsymbol{\sigma}}$ over the PDE parameters $\boldsymbol{\theta}$ and neural network parameters $\boldsymbol{\omega}$ (see Figure 2 b), given the noise parameters $\boldsymbol{\sigma} = [\sigma_u, \sigma_f]$.[1] Unlike the CNS approach, the B-PINN likelihood can be evaluated without explicitly solving the IBVP; instead, the $\mathcal{L}_{pde}$ term provides a *soft* enforcement of the PDE constraint. Given independent priors $p(\boldsymbol{\theta})$ and $p(\boldsymbol{\omega})$, the resulting B-PINN posterior takes the form $p(\boldsymbol{\theta}, \boldsymbol{\omega} \mid D_u, D_f) \propto \mathcal{L}_{\boldsymbol{\sigma}}(\boldsymbol{\theta}, \boldsymbol{\omega}; D_u, D_f)\, p(\boldsymbol{\theta})\, p(\boldsymbol{\omega})$. This posterior is analytically intractable, necessitating the use of approximate inference methods, two of which we detail below.

MCMC sampling can be used to perform inference in a B-PINN model, as with a CNS. However, the B-PINN posterior is defined over both the PDE parameters $\boldsymbol{\theta}$ and the neural network parameters $\boldsymbol{\omega}$, whereas the CNS posterior depends only on $\boldsymbol{\theta}$. In many practical applications the number of unknown PDE parameters is relatively small. For example, in systems biology (Duk et al., 2021), cardiac mechanics (Buoso et al., 2021), and haemodynamics (Paun et al., 2020), only a handful of parameters are typically inferred. By contrast, even modest PINN architectures involve thousands of trainable parameters $\boldsymbol{\omega}$. Consequently, MCMC sampling for a B-PINN must explore a much higher-dimensional space than for a CNS, which can degrade sampler efficiency and mixing (see for instance Figure C.6).

Variational inference offers a more computationally efficient alternative to MCMC by approximating the true posterior within a prescribed family of distributions. A widely used simplification is mean-field variational inference (MFVI), which assumes posterior independence between groups of latent variables (Jordan et al., 1999). In the PDE inverse problem setting this leads to the variational family $\mathcal{Q}_{\text{MF}} = \{q(\boldsymbol{\theta}, \boldsymbol{\omega}) = q_{\boldsymbol{\theta}}(\boldsymbol{\theta})\, q_{\boldsymbol{\omega}}(\boldsymbol{\omega})\}$, where $q_{\boldsymbol{\theta}}$ and $q_{\boldsymbol{\omega}}$ are probability densities.

When the PDE solution depends non-trivially on the parameter vector $\boldsymbol{\theta}$, the mean-field independence assumption becomes restrictive, as it prevents the variational distribution from capturing the posterior dependency between $\boldsymbol{\omega}$ and $\boldsymbol{\theta}$. This limitation of MFVI motivates the variational framework proposed in the following section, which is designed to capture the correct posterior structure.

4 FLOWPINNS

Given the discussion in Section 3, our goal is to develop a PIML framework that avoids explicitly solving the governing equations (as in a B-PINN), while performing probabilistic inference only over the PDE parameters $\boldsymbol{\theta}$ (as in a CNS). To understand how this might be achieved, consider the following question: why does the

[1] For notational clarity, we omit explicit treatment of initial and boundary conditions here. In practice, these constraints can be incorporated through an additional likelihood term or enforced exactly via transformations of the PINN output (Sukumar and Srivastava, 2022).

CNS-based likelihood in Figure 2 (a) depend only on $\boldsymbol{\theta}$, even though the solution function u is also unknown? The answer lies in the *well-posedness* of the IBVP: given a specified value of $\boldsymbol{\theta}$, the solution function u is unique and therefore deterministic. As a result, all uncertainty in the system can be attributed solely to the parameters $\boldsymbol{\theta}$. This implies that the CNS-based joint posterior over $\boldsymbol{\theta}$ and u admits the following factorisation

$$\begin{aligned} p(\boldsymbol{\theta}, u \mid D_u) &= p(\boldsymbol{\theta} \mid D_u) \cdot p(u \mid D_u, \boldsymbol{\theta}) \\ &= p(\boldsymbol{\theta} \mid D_u) \cdot \delta(u - u_{\boldsymbol{\theta}}) \end{aligned} \tag{9}$$

where $\delta(u - u_{\boldsymbol{\theta}})$ denotes a Dirac delta distribution centred at the unique solution $u_{\boldsymbol{\theta}}$ corresponding to the parameter value $\boldsymbol{\theta}$, as computed by the CNS.

Motivated by this factorised structure, we propose the following approximate posterior

$$q_{\boldsymbol{\phi},\boldsymbol{\omega}}(\boldsymbol{\theta}, u) = q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \cdot \delta(u - u_{\boldsymbol{\omega}}(\cdot, \boldsymbol{\theta})). \tag{10}$$

Here $q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$ is a flexible variational distribution modelled by an IAF with $\boldsymbol{\theta} = \boldsymbol{z}_K$, and $\delta(u - u_{\boldsymbol{\omega}}(\cdot, \boldsymbol{\theta}))$ is a Dirac distribution centred at a parameterised PINN $u_{\boldsymbol{\omega}} \colon \bar{\Omega} \times [0, T] \times \boldsymbol{\Theta} \to \mathbb{R}$, defined over the joint space of spatio-temporal and PDE parameter inputs (Cho et al., 2024). We refer to this architecture as a **flowPINN**.

Note that evaluating the Dirac delta in Eq. (10) amounts simply to fixing the input value of $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ in the parameterised PINN $u_{\boldsymbol{\omega}}$. This contrasts sharply with CNS-based posteriors, where evaluating the analogous Dirac delta in Eq. (9) requires an expensive call to the numerical solver. Furthermore, although the Dirac mapping $\delta(u - u_{\boldsymbol{\omega}}(\cdot, \boldsymbol{\theta}))$ is deterministic, the stochasticity in $\boldsymbol{\theta} \sim q_{\boldsymbol{\phi}}$ induces a distribution over possible PINN solutions (see Figure 1 for an illustration).

The flowPINN likelihood function $\mathcal{L}_{\boldsymbol{\omega},\boldsymbol{\sigma}}$ is shown in Figure 2 (c). Its structure is similar to that of a B-PINN, but with an important distinction: in a B-PINN, the PINN parameter vector $\boldsymbol{\omega}$ is treated as an input to the likelihood, which is *sampled* during inference. By contrast, a flowPINN treats $\boldsymbol{\omega}$ as a hyperparameter of the likelihood, which is *optimised* during inference.

The quality of the posterior approximation provided by $q_{\boldsymbol{\phi},\boldsymbol{\omega}}$ depends on the values of the flow parameters $\boldsymbol{\phi}$ and the PINN parameters $\boldsymbol{\omega}$. Furthermore, in most applications the noise parameters $\boldsymbol{\sigma}$ are also unknown *a priori*. To learn all three sets of parameters, we adopt the standard variational inference approach of maximising the evidence lower bound (ELBO) (Jordan et al., 1999):

$$\begin{aligned} \mathrm{ELBO}(\boldsymbol{\omega}, \boldsymbol{\phi}, \boldsymbol{\sigma}) = \mathbb{E}_{q_{\boldsymbol{\phi}}}[\log \mathcal{L}_{\boldsymbol{\omega},\boldsymbol{\sigma}}(\boldsymbol{\theta}; D_u, D_f)] \\ - \mathbb{D}_{\mathrm{KL}}[q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})]\,. \end{aligned} \tag{11}$$

The ELBO embodies the typical likelihood–prior trade-off in Bayesian inference (Blei et al., 2017). The first term is the expected log-likelihood under the variational distribution and acts as a data-fit term encouraging accurate predictions. The second term is the Kullback–Leibler (KL) divergence (Murphy, 2023, Ch. 5.1) between the variational posterior and the prior, which acts as a regularisation term.

Since $q_{\boldsymbol{\phi}}$ has been specified as a normalising flow, the ELBO can be re-written as an expectation with respect to the simple base distribution q_0 (see Appendix B.1). Although this expectation is analytically intractable, it can be approximated efficiently using samples from q_0.

4.1 Training

Unlike a standard variational inference problem, the ELBO objective in Eq. (11) contains trainable parameters not only in the variational distribution $q_{\boldsymbol{\phi}}$, but also in the likelihood approximation through the parameterised PINN $u_{\boldsymbol{\omega}}$. Previous work has shown that approximate posteriors with this structure can overfit the observation data and underestimate the true level of uncertainty (Yin and Zhou, 2018, Proposition 1), (Dabrowski and Pagendam, 2022, Appendix C). To mitigate this issue, we propose a three-stage training procedure for the flowPINN model.

In Stage One, a coarse approximation to the posterior over $\boldsymbol{\theta}$ is learned using randomly subsampled data. In Stage Two, the PINN is trained within the parameter support defined by the Stage One posterior while other parameters are held fixed. Finally, in Stage Three, the flow and likelihood parameters are learned using the full dataset with the PINN held fixed.

Stage One. In the first stage we jointly optimise the PINN parameters $\boldsymbol{\omega}$, the flow parameters $\boldsymbol{\phi}$, and the likelihood parameters $\boldsymbol{\sigma}$ by maximising a *subsampled* ELBO:

$$\boldsymbol{\omega}_1^\star, \boldsymbol{\phi}_1^\star, \boldsymbol{\sigma}_1^\star \;=\; \underset{\boldsymbol{\omega},\boldsymbol{\phi},\boldsymbol{\sigma}}{\arg\max}\, \tilde{\mathrm{ELBO}}(\boldsymbol{\omega}, \boldsymbol{\phi}, \boldsymbol{\sigma}; M). \tag{12}$$

Here $\tilde{\mathrm{ELBO}}(\cdot; M)$ denotes the ELBO evaluated on data subsamples of size $M < N_u$. Specifically, at each optimisation step we randomly sample M observation points (Eq. 7) and M collocation points (Eq. 8) and compute the ELBO using only these samples. Subsampling in this way injects stochasticity into the training routine and produces an implicit regularising (tempering) effect (Mandt et al., 2016, 2017; Smith and Le, 2017). For a flowPINN, this effect encourages the flow posterior $q_{\boldsymbol{\phi}_1^\star}$ to remain diffuse and cover a broader region of the PDE parameter space $\boldsymbol{\Theta}$.

Stage Two. In the second stage we refine the PINN parameters by maximum likelihood, with the flow pa-

rameters held fixed:

$$\boldsymbol{\omega}^\star = \arg\max_{\boldsymbol{\omega}} \mathbb{E}_{q_{\boldsymbol{\phi}_1^\star}} \left[\log \mathcal{L}_{\boldsymbol{\omega},\boldsymbol{\sigma}_1^*}(\boldsymbol{\theta} \mid D_u, D_f) \right]. \quad (13)$$

This step trains the PINN to provide accurate predictions in the region of $\boldsymbol{\Theta}$ to which $q_{\boldsymbol{\phi}_1^\star}$ assigns high probability.

Stage Three. In the final stage we use the full-data ELBO to update the flow and likelihood parameters, with the PINN held fixed:

$$\boldsymbol{\phi}^\star, \boldsymbol{\sigma}^\star = \arg\max_{\boldsymbol{\phi},\boldsymbol{\sigma}} \text{ ELBO}(\boldsymbol{\omega}^\star, \boldsymbol{\phi}, \boldsymbol{\sigma}). \quad (14)$$

A single application of Stage Two is sufficient because additional iterations after Stage Three has been completed would only retrain the PINN on a subset of the region where it is already accurate (see, for instance, Figure C.8), resulting in unnecessary computation without any expected improvement.

Choice of subsample size. The degree of regularisation in Stage One is controlled by the subsample size M. A smaller value of M reduces the influence of the dataset and therefore yields a more diffuse Stage One posterior $q_{\boldsymbol{\phi}_1^\star}$ which is closer to the prior (see Figure C.8). Appendix C.2 presents a sensitivity study on M, which shows that the final learned posterior is largely insensitive to its value, provided it is small enough for $q_{\boldsymbol{\phi}_1^\star}$ to cover the gold standard posterior. In our numerical experiments we set M to the smallest value (typically $M \approx 5$) that yields a stable estimate of the nugget noise hyperparameter σ_f (see Figure C.9).

4.2 Related work

PINN-based UQ methods. Bayesian approaches have become widely used for UQ in PINNs, most notably through B-PINNs, as discussed in Section 3.2. Various extensions have since been proposed, including adaptively re-weighted (Perez et al., 2023), domain-decomposed (Figueres et al., 2025), error-aware (Graf et al., 2022), and warm-started variants (Costa et al., 2024). These models have been applied across diverse domains such as materials science (Olivier et al., 2021; Fernández et al., 2023), power systems (Stock et al., 2023), wildfire modelling (Dabrowski et al., 2023), and fluid dynamics (Liu et al., 2024).

Beyond B-PINNs, other UQ-enabled PINN variants include Langevin dynamics-based approaches (Bai et al., 2021), physics-informed GANs (Yang et al., 2020; Meng et al., 2022). Ensemble methods can also offer strong UQ performance in physics-informed settings (Psaros et al., 2023). These include deep ensembles (Lakshminarayanan et al., 2017), which train multiple PINNs from different random initialisations, and snapshot ensembles (Huang et al., 2017), which extract multiple parameter samples from a single training run.

Alternative UQ methods. A range of machine learning approaches have been explored for parameter inference and UQ in PDEs. Statistical emulators (Gramacy, 2020), for example, have been used to address PDE inverse problems (Davies et al., 2019; Maso Talou et al., 2021), but typically require a large number of forward model evaluations to train the surrogate. Physics-informed Gaussian processes (GPs) have also been investigated (Raissi and Karniadakis, 2017; Long et al., 2022; Pförtner et al., 2024), although these methods are generally more effective for linear PDEs. A more recent direction is the statistical finite-element method (statFEM) (Girolami et al., 2021; Duffin et al., 2021; Akyildiz et al., 2022), which differs fundamentally from the PIML approaches discussed here because it requires the governing equations to be solved at each iteration of training.

Alternative parametrisations of Eq. (10). We parameterised the approximate variational posterior in Eq. (10) using a normalising flow and a PINN, which offers modelling flexibility and efficient ELBO-based training. However, the underlying principle of the variational approximation - whereby all uncertainty enters through the PDE parameters - is not limited to this choice.

For example, the flow could be replaced by a neural transport map (Ambrogioni et al., 2018), an implicit generative model (Liu and Wang, 2016), or a hierarchical variational model (Ranganath et al., 2016). Similarly, the PINN could be replaced with a neural operator (Li et al., 2024, 2021; Lu et al., 2021), physics-informed convolutional or graph neural networks (Fang, 2021; Gao et al., 2022), or a hybrid data/physics-informed architecture (Kelshaw and Magri, 2022). These alternatives may increase modelling flexibility and enable extensions to problems with, for example, spatially varying parameters, varying domains, or shock-like PDE behaviour, although often at the cost of increased complexity or loss of explicit density evaluation.

5 NUMERICAL EXPERIMENTS

We evaluate the performance of the flowPINN framework on several synthetic inverse problems in Sections 5.1 and 5.2. The experimental setup is summarised below, with full implementation details provided in Appendix B.

Table 1: Summary statistics for experimental results. Arrows indicate whether higher (↑) or lower (↓) values correspond to better performance. Values shown as mean ± one standard deviation. RMSE levels are rescaled for readability.

PDE	Poisson2D				Poisson3D			
Target	u		$\boldsymbol{\theta}$		u		$\boldsymbol{\theta}$	
Evaluation	RMSE ↓	NIP-G ↑	RMSE ↓	JSD ↓	RMSE ↓	NIP-G ↑	RMSE ↓	JSD ↓
Gold Std.	12.4 ± 2.5	100	12.2 ± 4.4	0.0	3.0 ± 0.3	100	11.2 ± 1.7	0.0
flowPINN	12.7 ± 2.0	98.0 ± 1.4	7.7 ± 2.1	0.62 ± 0.08	2.9 ± 0.3	97.4 ± 1.6	9.3 ± 1.0	0.10 ± 0.04
hmcPINN	15.6 ± 3.6	96.3 ± 2.7	12.7 ± 4.9	0.63 ± 0.13	3.4 ± 0.3	95.1 ± 1.3	7.7 ± 1.4	0.34 ± 0.08
mfviPINN	152.4 ± 9.4	64.0 ± 4.2	81.6 ± 5.2	2.83 ± 0.26	23.0 ± 2.8	82.5 ± 5.4	29.4 ± 17.3	10.10 ± 0.6
mcdPINN	27.0 ± 3.0	79.1 ± 5.7	17.9 ± 3.8	0.93 ± 0.15	12.2 ± 0.9	81.0 ± 5.7	38.2 ± 2.2	1.52 ± 0.11
densPINN	15.5 ± 4.2	87.3 ± 7.5	16.0 ± 5.7	1.85 ± 0.40	3.0 ± 0.4	86.0 ± 4.3	4.0 ± 1.4	1.81 ± 0.31
malaPIGP	14.8 ± 4.4	96.6 ± 2.5	11.9 ± 4.8	0.69 ± 0.13	-	-	-	-

PDE	FisherLike				Burgers			
Target	u		$\boldsymbol{\theta}$		u		$\boldsymbol{\theta}$	
Evaluation	RMSE ↓	MPL ↑	RMSE ↓	$\mathbb{H}$	RMSE ↓	MPL ↑	RMSE ↓	$\mathbb{H}$
flowPINN	5.5 ± 2.1	374 ± 167	3.9 ± 2.2	-4.7 ± 0.3	3.6 ± 0.5	69 ± 24	2.0 ± 1.0	-6.5 ± 0.3
hmcPINN	7.1 ± 1.4	108 ± 16	5.1 ± 0.6	-3.9 ± 0.3	6.3 ± 1.7	23 ± 2	4.4 ± 2.4	-5.8 ± 0.1
mfviPINN	72.7 ± 2.3	5 ± 0.1	73.6 ± 4.1	-6.7 ± 0.3	25.4 ± 3.2	1 ± 1	59.0 ± 5.4	-7.4 ± 0.1
mcdPINN	22.6 ± 7.3	43 ± 9.5	8.7 ± 2.5	-4.0 ± 0.3	7.2 ± 1.6	17 ± 2	6.8 ± 2.6	-7.2 ± 0.1
densPINN	10.8 ± 3.6	84 ± 42.6	7.4 ± 4.1	-5.2 ± 0.9	6.5 ± 1.1	55 ± 26	2.9 ± 0.9	-7.2 ± 0.4

Benchmark methods. The primary benchmark used to evaluate flowPINN is the Bayesian (B-)PINN. We consider three inference strategies for fitting B-PINNs: Hamiltonian Monte Carlo (HMC), mean-field variational inference (MFVI), and Monte Carlo dropout (MCD), which we denote as **hmcPINN**, **mfviPINN**, and **mcdPINN**, respectively. As a non-probabilistic baseline, we also consider deep ensembles, referred to as **densPINN**. For the two-dimensional Poisson equation experiment in Section 5.1, we also compare against physics-informed Gaussian processes (PIGPs) fit using the Metropolis-adjusted Langevin algorithm (MALA), denoted **malaPIGP**. Further details on all benchmark methods are provided in Appendix B.3.

Manual tuning for hmcPINN. The performance of hmcPINN is sensitive to the collocation noise hyperparameter σ_f. To mitigate this issue, we manually tune σ_f against the ground truth data to obtain best performance, the details of which are provided in Appendix B.7.7. Therefore the hmcPINN results we report can be expected to be better than would be available in practical applications, where such tuning is not possible. Recall that a flowPINN, by contrast, adaptively learns the noise hyperparameters during training.

Gold standard reference. For the experiments described in Section 5.1, we focused on the Poisson equation due to its tractability via the finite element method (FEM). This enables us to solve the associated inverse problems by embedding an FEM solver within the No-U-Turn Sampler (NUTS) MCMC algorithm (Hoffman et al., 2014), in which case the likelihood takes the form given in Figure 2 (a). We refer to this reference method as nutsFEM. Since nutsFEM explicitly enforces the governing PDE and operates with full Bayesian inference, we treat its results as the **gold standard** for benchmarking the performance of the machine learning methods.

Evaluation metrics. We assess the predictive performance and UQ of the PIML models using several complementary metrics, which are described in detail in Appendix B.7. Briefly, predictive accuracy for both the solution function u and the PDE parameters $\boldsymbol{\theta}$ is measured using the root mean squared error (**RMSE**). The quality of the predictive distributions for u is evaluated using the mean predictive likelihood (**MPL**). The normalised inner product of variances (**NIP-G**) is used to assess the calibration of uncertainty estimates for u against the gold standard nutsFEM results. To quantify the discrepancy between the estimated and gold standard posteriors in $\boldsymbol{\theta}$ space, the Jensen-Shannon Divergence (**JSD**) is employed. Additionally, we use the differential entropy $\mathbb{H}$ (Murphy, 2023, Ch. 5.2) to measure the overall uncertainty in the samples of $\boldsymbol{\theta}$.

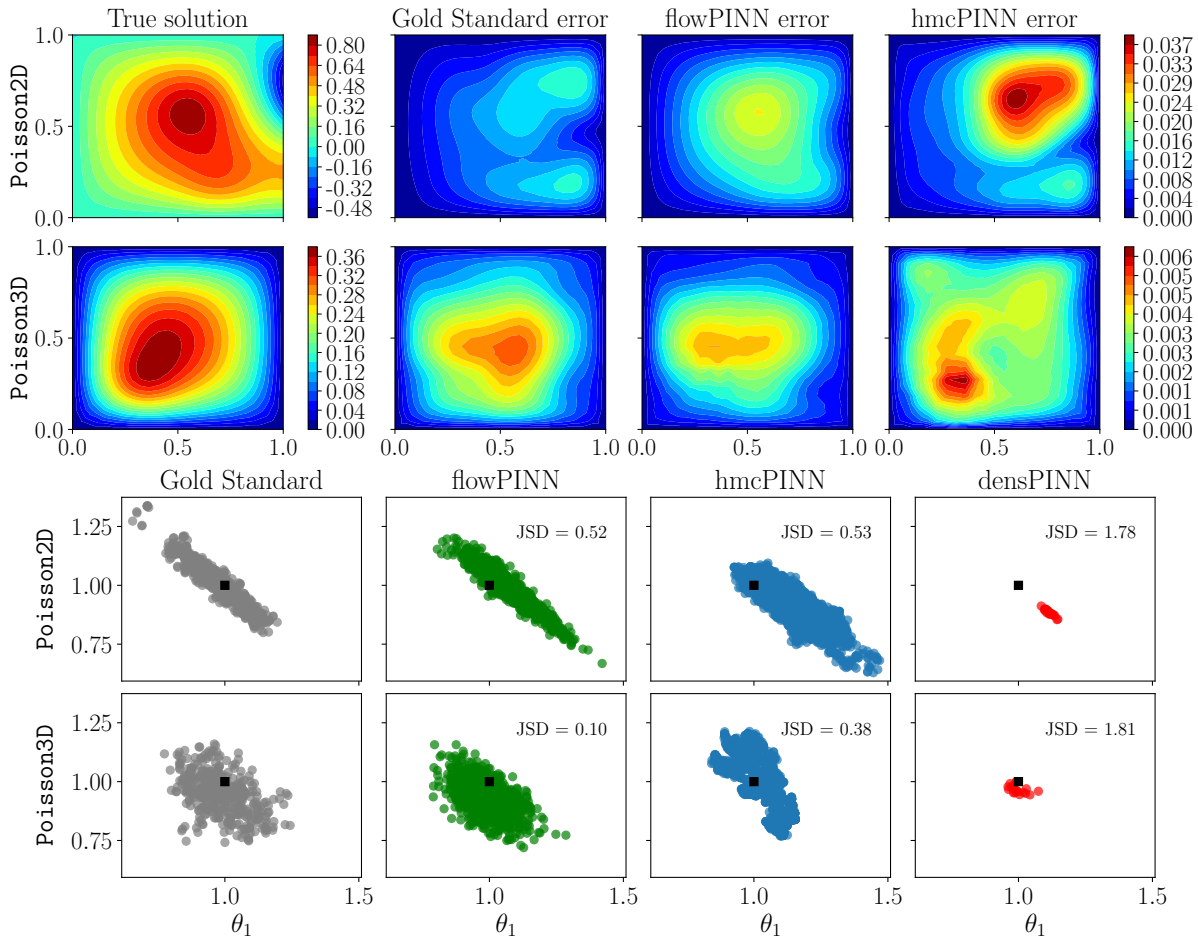


Figure 3: `Poisson2D` and `Poisson3D` results for flowPINN, hmcPINN, and densPINN, each evaluated on a single regenerated dataset. Gold standard nutsFEM results are provided for reference. The top two rows show density plots of the true solution and the corresponding prediction errors, while the bottom rows present scatter plots of posterior samples of the PDE parameters from each model. The ground truth $\boldsymbol{\theta}$ values are indicated as black squares. Overall, flowPINN provides both quantitatively and qualitatively closer agreement with the gold standard posterior for both datasets.

Implementation details. The flowPINN framework was implemented in Python using JAX (Bradbury et al., 2018) and Flax (Heek et al., 2020). The hmcPINN and densPINN baselines were implemented using the neuralUQ library (Zou et al., 2024) (v0.1.0-beta). Code and data required to reproduce the experiments are available at `www.github.com/dodaltuin/flowpinns`.

5.1 Poisson Equation

Our first set of experiments involve the Poisson equation (Eq. B.16) in $D = 2$ (`Poisson2D`) and $D = 3$ (`Poisson3D`) dimensions. As described in Appendix B.2, each experiment is repeated under multiple random regenerations of the data to estimate frequentist confidence intervals for the error metrics. The results are summarised in the top half of Table 1 and posterior plots are displayed in Figure 3.

For `Poisson2D`, flowPINN and hmcPINN achieve comparable performance relative to the gold standard nutsFEM results, both in terms of solution accuracy and posterior calibration. For `Poisson3D`, flowPINN more closely matches the gold standard, achieving a particularly low JSD value. We remark that hmcPINN exhibits poor posterior mixing for this problem (see Figure C.6), even after manual tuning of the collocation noise level σ_f and other MCMC hyperparameters. When compared to other PINN-based approaches, flowPINN consistently attains lower solution errors and better-calibrated uncertainty.

5.2 Nonlinear PDEs

We next consider two nonlinear PDEs: a `FisherLike` equation (Eq. B.17) and a `Burgers` equation (Eq. B.18). As in Section 5.1, we formulate parameter inverse prob-

Table 2: Training times (in minutes) for each experiment. The hmcPINN models are run on CPU (Intel® Xeon® Platinum 8276L @ 2.20,GHz) using the `neuralUQ` package (Zou et al., 2024). All other models are implemented in JAX (Bradbury et al., 2018) and run on GPU (NVIDIA RTX A6000).

Model	`Poisson2D`	`Poisson3D`	`FisherLike`	`Burgers`
flowPINN	7	9	5	11
hmcPINN	64	15	14	42
mfviPINN	2	5	1	9
mcdPINN	2	8	1	10

lems across multiple random regenerations of the data to enable frequentist uncertainty assessment. Summary results are reported in the bottom half of Table 1, with posterior samples shown in Figure C.5.

For both PDEs, flowPINN achieves slightly lower RMSE and higher MPL than hmcPINN, indicating improved predictive accuracy and better-calibrated uncertainty. Notably, hmcPINN exhibits good posterior mixing for these problems (see Figure C.7), suggesting that its results are close to a gold standard reference.

Among the other variational approaches, mcdPINN and mfviPINN show substantially lower accuracy than flowPINN in both experiments. DensPINN performs more comparably, particularly for the `Burgers` equation.

5.3 Experiment timings

The computational times required to train each of the probabilistic PINN-based models are provided in Table 2. The traditional VI approaches, mfviPINN and mcdPINN, are consistently the fastest to train, while hmcPINN is the slowest. FlowPINN exhibits intermediate computational cost. The hmcPINN models were implemented on CPU, as the `neuralUQ` package does not currently support GPU execution. Notably, the developers of the package report that HMC sampling can be faster on CPU due to more efficient random number generation (Psaros et al., 2023, Table 13).

5.4 Discussion

Across all experiments, flowPINN consistently demonstrates a favourable balance between predictive accuracy, uncertainty quantification, and computational efficiency. Compared with the variational baselines mfviPINN and mcdPINN, flowPINN achieves substantially lower errors as well as better-calibrated uncertainty estimates. FlowPINN also matches or exceeds the performance of hmcPINN, an asymptotically exact method, but with greater computational efficiency.

These results indicate that flowPINN preserves the accuracy of MCMC inference while retaining the efficiency of variational inference, making it an attractive method for solving PDE parameter inverse problems in settings where expensive forward solvers render standard MCMC approaches computationally prohibitive.

6 CONCLUSION

We have introduced flowPINNs, a variational framework that integrates a normalising flow with a PINN to provide uncertainty quantification in PDE parameter inference problems. Empirical results demonstrate that flowPINNs can match or exceed the performance of existing PINN-based UQ methods, with strong computational efficiency. Potential extensions and directions for future work are discussed in Appendix D.

Acknowledgements

This work has been funded by the Engineering and Physical Sciences Research Council (EPSRC) of the United Kingdom, grant reference number EP/T017899/1.

References

Abdulla, U. G. and Poteau, R. (2018). Identification of parameters in systems biology. *Mathematical Biosciences*, 305:133–145.

Akyildiz, Ö. D., Duffin, C., Sabanis, S., and Girolami, M. (2022). Statistical finite elements via langevin dynamics. *SIAM/ASA Journal on Uncertainty Quantification*, 10(4):1560–1585.

Ambrogioni, L., Güçlü, U., Güçlütürk, Y., Hinne, M., van Gerven, M. A., and Maris, E. (2018). Wasserstein variational inference. *Advances in Neural Information Processing Systems*, 31.

Arridge, S. R., Maass, P., Öktem, O., and Schönlieb, C.-B. (2019). Computational methods for inverse problems in imaging. *Acta Numerica*, 28:1–174.

Bai, H., Bhar, K., George, J., and Busart, C. (2021). Distributed bayesian parameter inference for physics-informed neural networks. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 2911–2916. IEEE.

Betancourt, M. (2017). A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs.

Buoso, S., Joyce, T., and Kozerke, S. (2021). Personalising left-ventricular biophysical models of the heart using parametric physics-informed neural networks. *Medical Image Analysis*, 71:102066.

Cabezas, A., Corenflos, A., Lao, J., Louf, R., Carnec, A., Chaudhari, K., Cohn-Gordon, R., Coullon, J., Deng, W., Duffield, S., et al. (2024). Blackjax: Composable bayesian inference in jax. *arXiv preprint arXiv:2402.10797*.

Cho, W., Jo, M., Lim, H., Lee, K., Lee, D., Hong, S., and Park, N. (2024). Parameterized physics-informed neural networks for parameterized pdes. *arXiv preprint arXiv:2408.09446*.

Costa, E. A., Rebello, C. M., Santana, V. V., and Nogueira, I. B. (2024). Physics-informed neural network uncertainty assessment through bayesian inference. *IFAC-PapersOnLine*, 58(14):652–657.

Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., and Piccialli, F. (2022). Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88.

Dabrowski, J. J. and Pagendam, D. E. (2022). Bayesian neural network inference via implicit models and the posterior predictive distribution. *arXiv preprint arXiv:2209.02188*.

Dabrowski, J. J., Pagendam, D. E., Hilton, J., Sanderson, C., MacKinlay, D., Huston, C., Bolt, A., and Kuhnert, P. (2023). Bayesian physics informed neural networks for data assimilation and spatio-temporal modelling of wildfires. *Spatial Statistics*, 55:100746.

Dalton, D., Lazarus, A., Gao, H., and Husmeier, D. (2024). Boundary constrained gaussian processes for robust physics-informed machine learning of linear partial differential equations. *Journal of Machine Learning Research*, 25(272):1–61.

Davies, V., Noè, U., Lazarus, A., Gao, H., Macdonald, B., Berry, C., Luo, X., and Husmeier, D. (2019). Fast parameter inference in a biomechanical model of the left ventricle by using statistical emulation. *Journal of the Royal Statistical Society. Series C, Applied Statistics*, 68(5):1555–1576.

Devore, J. L., Berk, K. N., Carlton, M. A., et al. (2012). *Modern mathematical statistics with applications*, volume 285. Springer.

Du, R., Zhou, T., and Pang, G. (2025). Forward and inverse problem solvers for reynolds-averaged navier–stokes equations with fractional laplacian. *Engineering Analysis with Boundary Elements*, 175:106193.

Duffin, C., Cripps, E., Stemler, T., and Girolami, M. (2021). Statistical finite elements for misspecified models. *Proceedings of the National Academy of Sciences*, 118(2):e2015006118.

Duk, M. A., Gursky, V. V., Samsonova, M. G., and Surkova, S. Y. (2021). Application of domain-and genotype-specific models to infer post-transcriptional regulation of segmentation gene expression in drosophila. *Life*, 11(11):1232.

Eymard, R., Gallouët, T., and Herbin, R. (2000). Finite volume methods. *Handbook of numerical analysis*, 7:713–1018.

Fan, J., Di Cristo, M., Jiang, Y., and Nakamura, G. (2010). Inverse viscosity problem for the navier–stokes equation. *Journal of mathematical analysis and applications*, 365(2):750–757.

Fang, Z. (2021). A high-efficient hybrid physics-informed neural networks based on convolutional neural network. *IEEE Transactions on Neural Networks and Learning Systems*, 33(10):5514–5526.

Fernández, J., Chiachío, J., Chiachío, M., Barros, J., and Corbetta, M. (2023). Physics-guided bayesian neural networks by abc-ss: Application to reinforced concrete columns. *Engineering Applications of Artificial Intelligence*, 119:105790.

Figueres, J. V., Vanderhaeghen, J., Bragone, F., Morozovska, K., and Shukla, K. (2025). $pinn–a domain decomposition method for bayesian physics-informed neural networks. *arXiv preprint arXiv:2504.19013*.

Gábor, A. and Banga, J. R. (2015). Robust and efficient parameter estimation in dynamic models of biological systems. *BMC systems biology*, 9(1):74.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR.

Gamerman, D. (2006). *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. Chapman & Hall/CRC.

Gao, H., Li, W., Cai, L., Berry, C., and Luo, X. (2015). Parameter estimation in a holzapfel–ogden law for healthy myocardium. *Journal of engineering mathematics*, 95:231–248.

Gao, H., Zahr, M. J., and Wang, J.-X. (2022). Physics-informed graph neural galerkin networks: A unified framework for solving pde-governed forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 390:114502.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. (2015). Made: Masked autoencoder for distribution

estimation. In *International conference on machine learning*, pages 881–889. PMLR.

Girolami, M., Febrianto, E., Yin, G., and Cirak, F. (2021). The statistical finite element method (statfem) for coherent synthesis of observation data and model predictions. *Computer Methods in Applied Mechanics and Engineering*, 375:113533.

Graf, O., Flores, P., Protopapas, P., and Pichara, K. (2022). Error-aware b-pinns: Improving uncertainty quantification in bayesian physics-informed neural networks. *arXiv preprint arXiv:2212.06965*.

Gramacy, R. B. (2020). *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. Chapman Hall/CRC, Boca Raton, Florida. `http://bobby.gramacy.com/surrogates/`.

Haghighat, E., Raissi, M., Moure, A., Gomez, H., and Juanes, R. (2021). Physics-informed neural networks for inverse problems in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 379:113741.

Hartmann, S. and Gilbert, R. R. (2018). Identifiability of material parameters in solid mechanics. *Archive of Applied Mechanics*, 88(1):3–26.

Heek, J., Levskaya, A., Oliver, A., Ritter, M., Rondepierre, B., Steiner, A., and van Zee, M. (2020). Flax: A neural network library and ecosystem for JAX.

Hoffman, M. D., Gelman, A., et al. (2014). The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623.

Hong, H., Eom, E., Lee, H., Choi, S., Choi, B., and Kim, J. K. (2024). Overcoming bias in estimating epidemiological parameters with realistic history-dependent disease spread dynamics. *Nature Communications*, 15(1):8734.

Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., and Weinberger, K. Q. (2017). Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*.

Jin, X., Cai, S., Li, H., and Karniadakis, G. E. (2021). Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, 426:109951.

Johnson, N. L., Kotz, S., and Balakrishnan, N. (1995). *Continuous Univariate Distributions, Volume 1*. John Wiley & Sons, New York, 2nd edition.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37:183–233.

Kaipio, J. and Somersalo, E. (2005). *Statistical and Computational Inverse Problems*. Springer.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. (2021a). Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440. Number: 6 Publisher: Nature Publishing Group.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. (2021b). Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440.

Kelshaw, D. and Magri, L. (2022). Physics-informed convolutional neural networks for corruption removal on dynamical systems. *arXiv preprint arXiv:2210.16215*.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. (2021). Fourier neural operator for parametric partial differential equations.

Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., and Anandkumar, A. (2024). Physics-informed neural operator for learning partial differential equations. *ACM / IMS J. Data Sci.*, 1(3).

Liu, H., Wang, Z., Deng, R., Wang, S., Meng, X., Xu, C., and Cai, S. (2024). Flow reconstruction with uncertainty quantification from noisy measurements based on bayesian physics-informed neural networks. *Physics of Fluids*, 36(11).

Liu, Q. and Wang, D. (2016). Stein variational gradient descent: A general purpose bayesian inference algorithm. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Logg, A., Mardal, K.-A., and Wells, G. (2012). *Automated solution of differential equations by the finite element method: The FEniCS book*, volume 84. Springer Science & Business Media.

Long, D., Wang, Z., Krishnapriyan, A., Kirby, R., Zhe, S., and Mahoney, M. (2022). AutoIP: A United Framework to Integrate Physics into Gaussian Processes. arXiv:2202.12316 [cs].

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. (2021). Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229.

Mandt, S., Hoffman, M., and Blei, D. (2016). A variational analysis of stochastic gradient algorithms. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 354–363, New York, New York, USA. PMLR.

Mandt, S., Hoffman, M. D., and Blei, D. M. (2017). Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 18(134):1–35.

Maso Talou, G. D., Babarenda Gamage, T. P., and Nash, M. P. (2021). Efficient ventricular parameter estimation using ai-surrogate models. *Frontiers in Physiology*, 12:732351.

Meng, X., Yang, L., Mao, Z., del Águila Ferrandis, J., and Karniadakis, G. E. (2022). Learning functional priors and posteriors from data and physics. *Journal of Computational Physics*, 457:111073.

Murphy, K. P. (2023). *Probabilistic Machine Learning: Advanced Topics*. MIT Press.

New, A., Villafañe-Delgado, M., and Shugert, C. (2024). Equation identification for fluid flows via physics-informed neural networks. *arXiv preprint arXiv:2408.17271*.

Olivier, A., Shields, M. D., and Graham-Brady, L. (2021). Bayesian neural networks for uncertainty quantification in data-driven materials modeling. *Computer methods in applied mechanics and engineering*, 386:114079.

Osi, A. and Ghaffarzadegan, N. (2024). Parameter estimation in behavioral epidemic models with endogenous societal risk-response. *PLOS Computational Biology*, 20(3):e1011992.

Pacheo, C., Dulikravich, G., Vesenjak, M., Borovinšek, M., Duarte, I., Jha, R., Reddy, S., Orlande, H., and Colaço, M. (2016). Inverse parameter identification in solid mechanics using bayesian statistics, response surfaces and minimization. *Technische Mechanik-European Journal of Engineering Mechanics*, 36(1-2):120–131.

Papamakarios, G., Pavlakou, T., and Murray, I. (2017). Masked autoregressive flow for density. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Paun, L. M., Colebank, M. J., Olufsen, M. S., Hill, N. A., and Husmeier, D. (2020). Assessing model mismatch and model selection in a bayesian uncertainty quantification analysis of a fluid-dynamics model of pulmonary blood circulation. *Journal of the Royal Society Interface*, 17(173):20200886.

Peherstorfer, B., Willcox, K., and Gunzburger, M. (2018). Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *Siam Review*, 60(3):550–591.

Perez, S., Maddu, S., Sbalzarini, I. F., and Poncet, P. (2023). Adaptive weighting of bayesian physics informed neural networks for multitask and multiscale forward and inverse problems. *Journal of Computational Physics*, 491:112342.

Pförtner, M., Steinwart, I., Hennig, P., and Wenger, J. (2024). Physics-informed gaussian process regression generalizes linear pde solvers.

Prince, S. J. (2023). *Understanding deep learning*. MIT press.

Psaros, A. F., Meng, X., Zou, Z., Guo, L., and Karniadakis, G. E. (2023). Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. *Journal of Computational Physics*, 477:111902.

Rackauckas, C., Ma, Y., Martensen, J., Warner, C., Zubov, K., Supekar, R., Skinner, D., Ramadhan, A., and Edelman, A. (2020). Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*.

Raissi, M. and Karniadakis, G. E. (2017). Machine learning of linear differential equations using gaussian processes. *Journal of Computational Physics*, 348:683–693.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017). Machine learning of linear differential equations using Gaussian processes. *Journal of Computational Physics*, 348:683–693.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707.

Ranganath, R., Tran, D., and Blei, D. (2016). Hierarchical variational models. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 324–333, New York, New York, USA. PMLR.

Rappel, H., Beex, L. A., Hale, J. S., Noels, L., and Bordas, S. (2020). A tutorial on bayesian inference to identify material parameters in solid mechanics. *Archives of Computational Methods in Engineering*, 27(2):361–385.

Ryan, W., Taylor-LaPole, A., Olufsen, M., Husmeier, D., and Vyshemirskiy, V. (2026). Physics-informed emulation of systemic circulation for fast parameter estimation and uncertainty quantification. *International Journal for Numerical Methods in Biomedical Engineering*, 42(2):e70147.

Sabat, L. and Kundu, C. K. (2020). History of finite element method: a review. *Recent Developments in Sustainable Infrastructure: Select Proceedings of ICRDSI 2019*, pages 395–404.

Smirnova, A. and Chowell, G. (2017). A primer on stable parameter estimation and forecasting in epidemiology by a problem-oriented regularized least squares algorithm. *Infectious Disease Modelling*, 2(2):268–275.

Smith, G. D. (1985). *Numerical solution of partial differential equations: finite difference methods*. Oxford university press.

Smith, S. L. and Le, Q. V. (2017). A bayesian perspective on generalization and stochastic gradient descent. *arXiv preprint arXiv:1710.06451*.

Stock, S., Stiasny, J., Babazadeh, D., Becker, C., and Chatzivasileiadis, S. (2023). Bayesian physics-informed neural networks for robust system identification of power systems. In *2023 IEEE Belgrade PowerTech*, pages 1–6. IEEE.

Strauss, W. A. (2007). *Partial differential equations: An introduction*. John Wiley & Sons.

Stuart, A. M. (2010). Inverse problems: A bayesian perspective. *Acta Numerica*, 19:451–559.

Sukumar, N. and Srivastava, A. (2022). Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 389:114333.

Tarantola, A. (2005). *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics.

Verma, A., Jiwari, R., and Koksal, M. E. (2014). Analytic and numerical solutions of nonlinear diffusion equations via symmetry reductions. *Advances in Difference Equations*, (1):229.

Xifara, T., Sherlock, C., Livingstone, S., Byrne, S., and Girolami, M. (2014). Langevin diffusions and the metropolis-adjusted langevin algorithm. *Statistics & Probability Letters*, 91:14–19.

Xun, X., Cao, J., Mallick, B., Maity, A., and Carroll, R. J. (2013). Parameter estimation of partial differential equation models. *Journal of the American Statistical Association*, 108(503):1009–1020.

Yang, L., Meng, X., and Karniadakis, G. E. (2021). B-PINNs: Bayesian physics-informed neural networks for forward and inverse pde problems. *Journal of Computational Physics*, 429:109927.

Yang, L., Zhang, D., and Karniadakis, G. E. (2020). Physics-informed generative adversarial networks for stochastic differential equations. *SIAM Journal on Scientific Computing*, 42(1):A292–A317.

Yin, M. and Zhou, M. (2018). Semi-implicit variational inference. In *International conference on machine learning*, pages 5660–5669. PMLR.

Zienkiewicz, O. C., Taylor, R. L., and Zhu, J. Z. (2005). *The finite element method: its basis and fundamentals*. Elsevier.

Zou, Z., Meng, X., Psaros, A. F., and Karniadakis, G. E. (2024). Neuraluq: A comprehensive library for uncertainty quantification in neural differential equations and operators. *SIAM Review*, 66(1):161–190.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes/No/Not Applicable]: Yes - we have provided a detailed explanation of our proposed algorithm in Section 4, with further details provided in the appendix.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes/No/Not Applicable]: No - we did not provide a formal analysis of the complexity of our proposed approach, as it makes use of two standard architectures (normalising flows and PINNs).
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes/No/Not Applicable]: Yes - we have provided a GitHub link to the supporting code and data.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes/No/Not Applicable]: Not Applicable.
   (b) Complete proofs of all theoretical results. [Yes/No/Not Applicable]: Not Applicable.
   (c) Clear explanations of any assumptions. [Yes/No/Not Applicable]: Not Applicable.

3. For all figures and tables that present empirical results, check if you include:

(a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes/No/Not Applicable]: Yes.

(b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes/No/Not Applicable]: Yes - see Appendix B.1.

(c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes - see Appendix B.1.

(d) A description of the computing infrastructure used. [Yes/No/Not Applicable]: Yes - see the paragraph "Implementation Details" at the start of Section 5. .

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

   (a) Citations of the creator If your work uses existing assets. [Yes/No/Not Applicable]: Yes - see the paragraph "Implementation Details" at the start of Section 5.

   (b) The license information of the assets, if applicable. [Yes/No/Not Applicable]: Yes.

   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes/No/Not Applicable]: Not Applicable.

   (d) Information about consent from data providers/curators. [Yes/No/Not Applicable]: Not Applicable.

   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Yes/No/Not Applicable]: Not Applicable.

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

   (a) The full text of instructions given to participants and screenshots. [Yes/No/Not Applicable]: Not Applicable.

   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Yes/No/Not Applicable]: Not Applicable.

   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Yes/No/Not Applicable]: Not Applicable.

Supplementary Materials

A Solution space uncertainty quantification

For a fixed PDE parameter value $\boldsymbol{\theta}$, a flowPINN yields a deterministic solution function $\boldsymbol{\theta} \mapsto u_{\boldsymbol{\omega}}(\cdot, \boldsymbol{\theta})$. However, the parameters themselves are random, with $\boldsymbol{\theta} \sim q_{\boldsymbol{\phi}}$. As a result, the flowPINN solution is *stochastic* when $\boldsymbol{\theta}$ is not conditioned on, and follows a distribution that is induced by coupling the normalising flow $q_{\boldsymbol{\phi}}$ with the PINN $u_{\boldsymbol{\omega}}$. Sampling a function from this induced distribution is straightforward. Firstly, draw a parameter sample $\boldsymbol{\theta} \sim q_{\boldsymbol{\phi}}$. Then, evaluate the corresponding prediction function $u_{\boldsymbol{\omega}}(\cdot, \boldsymbol{\theta})$ by fixing $\boldsymbol{\theta}$ as input to the PINN. Repeating this process S times produces an ensemble $u_{\boldsymbol{\omega}}(\cdot, \boldsymbol{\theta}_1),\ u_{\boldsymbol{\omega}}(\cdot, \boldsymbol{\theta}_2),\ \ldots,\ u_{\boldsymbol{\omega}}(\cdot, \boldsymbol{\theta}_S)$, which provides an empirical approximation of the solution distribution and enables uncertainty quantification in function space.

Figure 1 illustrates this sampling procedure for the following one-dimensional reaction-diffusion equation:

$$-\theta_1 u_{xx}(x) + \theta_2 u(x) = f(x), \quad x \in \Omega = (0, 1), \tag{A.15}$$

where $\boldsymbol{\theta} = (\theta_1, \theta_2) = (0.1, 2.)$, $u(0) = u(1) = 0$ and $f(x) = 0.1(6x - 2) + 2x^2(1 - x)$. Taken together, these conditions imply that $u(x) = x^2(1 - x)$. We set up a simple flowPINN inference problem for this PDE using a dataset D_u of $N_u = 4$ noise corrupted observations of u, and performed training as described in Section 4.1. The left hand panel of Figure 1 displays $S = 1000$ samples from the normalising flow $q_{\boldsymbol{\phi}}$ after training. Each sample $\boldsymbol{\theta}_i$ is then mapped to a flowPINN solution function $\boldsymbol{\theta}_i \to u_{\boldsymbol{\omega}}(\cdot, \boldsymbol{\theta}_i)$ (centre panel). This can in turn be mapped to forcing space by applying the PDE operator in Eq. (A.15) to $u_{\boldsymbol{\omega}}(\cdot, \boldsymbol{\theta}_i)$ (right panel).

Note from Figure 1 that there is significant uncertainty in the flowPINN samples for the PDE parameter $\boldsymbol{\theta}$ and solution function u. This is because $\boldsymbol{\theta}$ and u are not known *a-priori*, and there is insufficient information in the sparse, noise-corrupted observation data to learn their values exactly.

By contrast, there is virtually no uncertainty in the flowPINN samples of the right-hand side forcing term f. This highlights a key aspect of the flowPINN framework. Recall that f is known *a-priori*. Therefore, to be consistent with the known physics of the problem, each parameter-function sample pair $(\boldsymbol{\theta}_i, u_{\boldsymbol{\omega}}(\cdot, \boldsymbol{\theta}_i))$ should satisfy the PDE (Eq. A.15 in this example, Eq. 1 in general). In other words, $\mathcal{F}_{\boldsymbol{\theta}_i}[u_{\boldsymbol{\omega}}](\cdot, \boldsymbol{\theta}_i)$ should equal $f(\cdot)$ for all $\boldsymbol{\theta}_i \sim q_{\boldsymbol{\phi}}$. This consistency requirement is (softly) enforced by a flowPINN through the fine-tuning of the PINN parameters $\boldsymbol{\omega}$ in Stage Two training (see Eq. 13).

B Experimental setup and implementation details

B.1 Re-expression of ELBO in terms of base distribution q_0

The ELBO in Eq. (11) can be re-expressed in terms of an expectation with respect the base distribution as follows:

$$\begin{aligned}
\mathrm{ELBO}(\boldsymbol{\omega}, \boldsymbol{\phi}, \boldsymbol{\sigma}) &= \mathbb{E}_{q_{\boldsymbol{\phi}}}[\log \mathcal{L}_{\boldsymbol{\omega},\boldsymbol{\sigma}}(\boldsymbol{\theta}; D_u, D_f)] - \mathbb{D}_{\mathrm{KL}}[q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})] \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log p(\boldsymbol{\theta}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) + \log \mathcal{L}_{\boldsymbol{\omega},\boldsymbol{\sigma}}(\boldsymbol{\theta}; D_u, D_f)\right] \\
&= \mathbb{E}_{q_0} \log\left[p(\boldsymbol{z}_K) \mathcal{L}_{\boldsymbol{\omega},\boldsymbol{\sigma}}(\boldsymbol{z}_K; D_u, D_f)\right] - \mathbb{E}_{q_0}\left[\log q_{\boldsymbol{\phi}}(\boldsymbol{z}_K)\right] \\
&= \mathbb{E}_{q_0} \log\left[p(\boldsymbol{z}_K) \mathcal{L}_{\boldsymbol{\omega},\boldsymbol{\sigma}}(\boldsymbol{z}_K; D_u, D_f)\right] + \mathbb{E}_{q_0}\left[\sum_{k=1}^{K} \log \left| \det\left(\frac{\partial \boldsymbol{g}_k}{\partial \boldsymbol{z}_{k-1}}\right)\right|\right],
\end{aligned}$$

where we have set $\boldsymbol{z}_k = \boldsymbol{\theta}$. The final line holds by Eq. (4), ignoring the log density of the prior on $\boldsymbol{z}_0$, which is a constant.

B.2 PDE Setup

Poisson Equation. Our initial series of numerical experiments focused on the Poisson equation, the fundamental example of an elliptic PDE:

$$-\nabla \cdot (\boldsymbol{\theta} \odot \nabla u(\boldsymbol{x})) = f(\boldsymbol{x}),\ \boldsymbol{x} \in \Omega. \tag{B.16}$$

It serves as a canonical test case due to its well-understood mathematical properties, availability of efficient numerical solutions and its relevance in modelling steady-state phenomena such as heat distribution, electrostatics, and gravitational potentials. As discussed in Section 5.1, we considered experiments involving a 2D Poisson equation (`Poisson2D`), and a 3D Poisson equation (`Poisson3D`), with $\boldsymbol{\theta} = \mathbf{1}^D$ and $\Omega = (0,1)^D$ in each case.

For `Poisson2D`, we used 10 random regenerations of the observation dataset D_u (see Eq. 7) for the inference problem, from which the error bars in Table 1 are derived. For each dataset, the right hand side function f in Eq. (B.16) was defined as a sum of three localised Gaussian source terms, each with randomly assigned location and magnitude. Dirichlet boundary conditions were specified as $u(\boldsymbol{x}) = \sin(\pi x_2)$ for $x_1 = 1$, and $u(\boldsymbol{x}) = 0$ on the remaining boundaries. The dataset D_u was created with 25 noisy observations (10% observation noise) of the solution u at randomly selected input locations, and D_f (see Eq. 8) was created with 1024 observations of f at input locations chosen using a space-filling design. In each case, the ground truth solution function was evaluated using the finite-element method (FEM), implemented in FEniCS (Logg et al., 2012).

The `Poisson3D` model followed a similar setup. We considered 5 random regenerations of the observation datset D_u, with f defined as the sum of two randomly positioned and weighted source terms in each case. Homogeneous Dirichlet boundary conditions ($u(\boldsymbol{x}) = 0$) were imposed across the entire boundary. Observation datasets D_u were created with 50 noisy observations of u (10% observation noise) at randomly selected input locations, while D_f was specified with 2500 observations of f at space-filling input locations. Again, FEniCS was used to evaluate the ground truth solution function for each dataset.

Fisher-like Equation. We considered the nonlinear Fisher-like PDE from Verma et al. (2014):

$$\frac{\partial u}{\partial t} - \theta_1 \frac{\partial^2 u}{\partial x^2} - \theta_2 u^2(1-u) = f = 0, \quad (x,t) \in (-5,5) \times (0,10], \tag{B.17}$$

with ground truth parameters $\boldsymbol{\theta} = (\theta_1, \theta_2) = (1,1)$ and corresponding closed-form solution from Eq. (35) of Verma et al. (2014). Dirichlet conditions were applied in both space and time. We considered 10 random regenerations of the dataset D_u, where in each case the location and noise values (5% observation noise) of 25 solution space observations were randomly sampled. The error bars reported in Table 1 are taken with respect this data resampling. $N_f = 250$ observations of $f = 0$ were used to the create dataset D_f, the input locations of which were randomly sampled from the spatio-temporal domain.

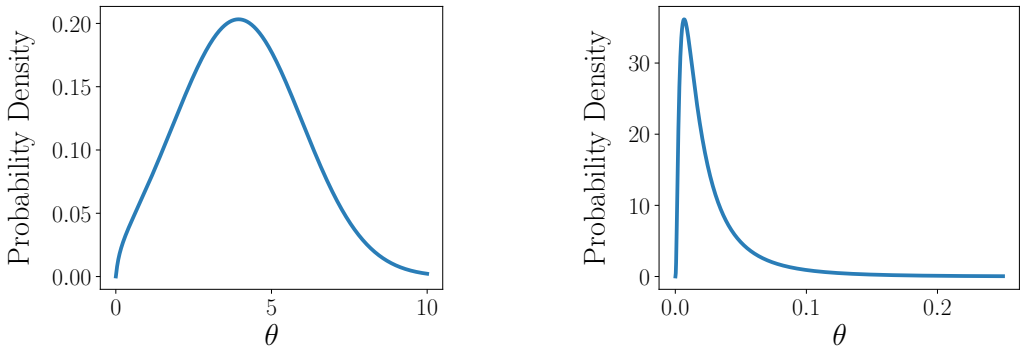
Burger's Equation. We used the well-known viscous Burger's equation from Raissi et al. (2019, Eq. A.1):

$$\frac{\partial u}{\partial t} - \theta_1 \frac{\partial^2 u}{\partial x^2} + \theta_2 u \frac{\partial u}{\partial x} = f = 0, \quad (x,t) \in (-1,1) \times (0,1], \tag{B.18}$$

with initial condition $u(x,0) = -\sin(\pi x)$ and Dirichlet boundaries $u(0,t) = u(1,t) = 0$. We adopted the solution data provided by Raissi et al. (2019), which corresponds to the parameter value $\boldsymbol{\theta} = (\theta_1, \theta_2) = (0.01/\pi, 1)$. To allow for frequentist error bars to be computed (see Table 1), we considered 5 random regenerations of the observation dataset D_u (in the same manner as the Fisher-like equation above). $N_f = 2500$ observations of $f = 0$ were used to the create dataset D_f, also with randomly sampled input locations.

B.2.1 PDE parameter prior distributions

For all experiments, the individual elements $\theta_1, \ldots, \theta_{N_p}$ in the PDE parameter vector $\boldsymbol{\theta}$ were each assigned independent Softplus-Normal(4,2) prior distributions. The density plot for this distribution is provided in Figure B.4 (a). It provides a non-negative distribution over parameter values less than 10, while avoiding the heavy tails of the Log-Normal distribution. An exception was made for the diffusion parameter θ_1 in Eq. (B.18), which we assumed followed a Softplus-Normal(-4,1) distribution, which is illustrated in Figure B.4 (b).



(a) Softplus(θ) $\sim \mathcal{N}(4, 2)$

(b) Softplus(θ) $\sim \mathcal{N}(-4, 1)$

Figure B.4: Density plots of the Softplus-Normal prior distributions used for the PDE parameters in the numerical experiments. These priors ensure positivity in the parameters while avoiding the heavy-tail of a Log-Normal distribution. Further discussion is provided in Appendix B.2.1.

B.3 Benchmark Methods

In Section 5, we benchmark the performance of flowPINN against several existing physics-informed machine learning (PIML) frameworks. We describe these benchmark methods in detail below.

B.4 Bayesian Physics-Informed Neural Networks

Bayesian Physics-Informed Neural Networks (B-PINNs) extend classical PINNs by placing probabilistic priors over unknown quantities in the model (Yang et al., 2021). Instead of computing point estimates of the PDE parameters $\boldsymbol{\theta}$ and network parameters $\boldsymbol{\omega}$ through optimisation of Eq. (2) like a classical PINN, a B-PINN aims to infer a posterior distribution over their values.

As discussed in Section 3.2, for the parameter inverse problems considered in this work, this posterior distribution takes the form

$$p(\boldsymbol{\theta}, \boldsymbol{\omega} \mid D_u, D_f) \propto \mathcal{L}_{\boldsymbol{\sigma}}(\boldsymbol{\theta}, \boldsymbol{\omega}; D_u, D_f)\, p(\boldsymbol{\theta}) p(\boldsymbol{\omega}), \tag{B.19}$$

where D_u is the solution-space observation dataset (see Eq. 7) and D_f is the PDE collocation dataset (see Eq. 8). The vector $\boldsymbol{\sigma} = [\sigma_u, \sigma_f]$ contains the observation noise hyperparameters, $\mathcal{L}_{\boldsymbol{\sigma}}$ is the likelihood function (see Figure 2 b), while $p(\boldsymbol{\theta})$ and $p(\boldsymbol{\omega})$ specify the prior distributions over the unknown parameters. The specific form of the PDE parameter prior for each experiment is detailed in Appendix B.2.1. For the network parameters $\boldsymbol{\omega}$, we assign independent standard normal priors to the individual weights and biases.

Exact inference in this posterior distribution is intractable, which necessitates the use of approximate Bayesian inference techniques. In this work, we consider three such strategies, which are summarised below.

B.4.1 Hamiltonian Monte Carlo

Markov chain Monte Carlo (MCMC) methods provide asymptotically exact Bayesian inference by drawing samples from the joint posterior distribution in Eq. (B.19). In particular, we employ the Hamiltonian Monte Carlo (HMC) algorithm, the use of which is standard in the B-PINN literature (Yang et al., 2021; Stock et al., 2023). HMC leverages gradients of the log-posterior to guide exploration of the parameter space, resulting in more efficient sampling compared to random-walk proposals (Betancourt, 2017). Results obtained using a B-PINN trained via HMC are referred to as **hmcPINN** in Section 5.

We found that the performance of hmcPINN was highly sensitive to the choice of the collocation noise hyperparameter σ_f. The results reported in Section 5 were obtained by manual tuning of σ_f, further details of which are provided in Appendix B.7.7.

We also experimented with the Metropolis-Adjusted Langevin Algorithm (MALA) (Xifara et al., 2014) as a potentially more efficient alternative to HMC, but were unable to achieve satisfactory posterior mixing. We additionally tested the No-U-Turn Sampler (NUTS) (Hoffman et al., 2014), which we found to be prohibitively slow in this setting. In particular, identifying U-turn points in the high-dimensional and multimodal posterior landscape of a B-PINN typically required a large number of leapfrog steps, leading to excessive computational cost.

B.4.2 Mean-field Variational Inference

Mean-field variational inference (MFVI) approximates the intractable posterior in Eq. (B.19) with a factorised variational distribution,

$$q_{\boldsymbol{\psi}}(\boldsymbol{\omega}, \boldsymbol{\theta}) = q_{\psi_{\boldsymbol{\omega}}}(\boldsymbol{\omega})\, q_{\psi_{\boldsymbol{\theta}}}(\boldsymbol{\theta}), \tag{B.20}$$

in which each factor is an isotropic Gaussian, with mean and scale parameters denoted $\boldsymbol{\psi} = [\boldsymbol{\psi}_{\boldsymbol{\omega}}, \boldsymbol{\psi}_{\boldsymbol{\theta}}]$. The variational parameters are obtained by maximisation of the following ELBO

$$\mathrm{ELBO}(\boldsymbol{\psi}, \boldsymbol{\sigma}) = \mathbb{E}_{q_{\boldsymbol{\psi}}}[\log \mathcal{L}_{\boldsymbol{\sigma}}(\boldsymbol{\theta}, \boldsymbol{\omega}; D_u, D_f)] - \mathbb{D}_{\mathrm{KL}}[q_{\psi_{\boldsymbol{\omega}}}(\boldsymbol{\omega}) \,\|\, p(\boldsymbol{\omega})] - \mathbb{D}_{\mathrm{KL}}[q_{\psi_{\boldsymbol{\theta}}}(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})]\,. \tag{B.21}$$

We refer to results obtained using MFVI as **mfviPINN** in Section 5. MFVI scales well to larger problems, but the conditional independence assumption between $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$ can limit its accuracy in problems with strong $\boldsymbol{\theta}$-$\boldsymbol{\omega}$ coupling.

B.4.3 Monte Carlo Dropout

Monte Carlo dropout (MCD) can be interpreted within a variational inference framework, in which uncertainty in neural network parameters is represented implicitly through stochastic dropout masks rather than through an explicit parametric variational distribution (Gal and Ghahramani, 2016).

In this work, we adopted a hybrid variational formulation that parallels mean-field variational inference (MFVI). Specifically, a parametric Gaussian variational approximation was used for the PDE parameters $\boldsymbol{\theta}$, while the posterior over the neural network parameters $\boldsymbol{\omega}$ was approximated using MCD. We therefore considered a factorised variational posterior of the form

$$q_{\boldsymbol{\psi}}(\boldsymbol{\omega}, \boldsymbol{\theta}) = q_{\mathrm{MCD}}(\boldsymbol{\omega})\, q_{\psi_{\boldsymbol{\theta}}}(\boldsymbol{\theta}), \tag{B.22}$$

where $q_{\psi_{\boldsymbol{\theta}}}(\boldsymbol{\theta})$ is an isotropic Gaussian with variational parameters $\psi_{\boldsymbol{\theta}}$, and $q_{\mathrm{MCD}}(\boldsymbol{\omega})$ denotes the implicit variational distribution induced by applying dropout to the network weights. As in MFVI, the variational parameters are learned by maximising an evidence lower bound (ELBO), which in this case can be written as

$$\mathrm{ELBO}(\boldsymbol{\psi}_{\boldsymbol{\theta}}, \boldsymbol{\sigma}) = \mathbb{E}_{q_{\boldsymbol{\psi}}(\boldsymbol{\omega}, \boldsymbol{\theta})}[\log \mathcal{L}_{\boldsymbol{\sigma}}(\boldsymbol{\theta}, \boldsymbol{\omega}; D_u, D_f)] - \mathbb{D}_{\mathrm{KL}}[q_{\psi_{\boldsymbol{\theta}}}(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})] - \mathbb{D}_{\mathrm{KL}}[q_{\mathrm{MCD}}(\boldsymbol{\omega}) \,\|\, p(\boldsymbol{\omega})]\,. \tag{B.23}$$

We approximated the KL term involving $q_{\mathrm{MCD}}(\boldsymbol{\omega})$ using standard dropout regularisation. Monte Carlo estimates of the expectation in Eq. (B.23) were obtained by sampling both the PDE parameters $\boldsymbol{\theta}$ and the dropout masks during training. Predictive uncertainty was then estimated by performing multiple stochastic forward passes with dropout enabled.

Results obtained using this approach are referred to as **mcdPINN** in Section 5. Compared with MFVI, MCD removes the need to specify an explicit parametric posterior over $\boldsymbol{\omega}$. However, it still suffers from the inherent modelling limitation induced by the conditional independence assumption between $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$.

We also experimented with more expressive variational families for the PDE parameter posterior $q_{\psi_{\boldsymbol{\theta}}}(\boldsymbol{\theta})$, including normalising flows and full-rank Gaussian approximations. In both cases, however, the learned posterior consistently collapsed to a distribution that was effectively isotropic Gaussian.

B.5 Deep Ensembles

Deep ensembles provide a non-Bayesian approach to UQ in PINNs (Zou et al., 2024). The method involves training multiple PINNs independently, each with different random initialisations and, optionally, different data subsampling. Each individual ensemble member produces a deterministic estimate of the solution and PDE

parameters, while the empirical distribution of the entire ensemble captures uncertainty arising from optimisation non-convexity and data sparsity. Deep ensembles are straightforward to implement and often provide robust UQ estimates, however, they do not correspond to a formal Bayesian posterior.

For the parameter inverse problems in Section 5, each ensemble member was trained by minimising the objective function in Eq. (2). Uncertainty was then quantified by aggregating predictions across the ensemble. We refer to the results obtained using this approach as **densPINN**.

B.6 Physics-Informed Gaussian Processes

Physics-informed Gaussian processes (PIGPs) provide a probabilistic framework for modelling linear PDEs (Raissi et al., 2017). In this approach, a Gaussian process (GP) prior is placed over the solution function u, which induces a corresponding GP prior over the forcing term f through application of the linear differential operator. Observations of the solution and enforcement of the PDE constraints are incorporated through Gaussian likelihoods, yielding a closed-form posterior distribution.

We applied PIGPs to the Poisson equation experiments described in Section 5.1. Inference was performed using the Metropolis-adjusted Langevin algorithm (MALA) (Xifara et al., 2014), implemented with the BlackJAX library (Cabezas et al., 2024). We refer to these results as **malaPIGP**. Dirichlet boundary conditions were enforced explicitly in both the mean and kernel functions of the PIGP using the framework of Dalton et al. (2024).

PIGPs provide well-calibrated uncertainty quantification, but their practical applicability is typically limited to moderate dataset sizes due to the cubic computational cost of GP inference. Although extensions to nonlinear PDEs have been proposed (for example Long et al., 2022), we do not pursue them here, as our primary focus is on PINN-based approaches.

B.7 Results evaluation metrics

In each numerical experiment, the **ground truth** PDE parameters $\boldsymbol{\theta}$ and solution function u are known, allowing the RMSE (Appendix B.7.1) and MPL (Appendix B.7.2) error metrics to be computed directly.

For the Poisson equation experiments in Section 5.1, we also solved the corresponding inverse problems using nutsFEM. We treat the nutsFEM posterior as the **gold standard**, since this method enforces the governing equations by explicitly solving the PDE at every MCMC iteration.

To evaluate the machine learning models against this reference, we report NIP-G (Appendix B.7.3) and JSD (Appendix B.7.4). It is important to note that the gold standard posterior does not necessarily yield the lowest pointwise error relative to the ground truth parameters. For example, in the `Poisson2D` experiment, flowPINN achieves lower RMSE values than nutsFEM because the learned posterior is slightly more concentrated around the ground truth PDE parameter value.

B.7.1 Root mean squared error (RMSE)

The *root mean squared error (RMSE)* metric evaluates the pointwise prediction accuracy of a model against the ground truth value. In solution space, RMSE is defined as

$$\text{RMSE} = \sqrt{\frac{1}{N_\star}\sum_{i=1}^{N_\star}\left[u(\boldsymbol{x}_\star^{(i)}) - \hat{u}(\boldsymbol{x}_\star^{(i)})\right]^2}$$

where u is the true solution function, $\hat{u}$ the prediction of the model, and the test points $\boldsymbol{x}_\star^{(1)}, \ldots, \boldsymbol{x}_\star^{(N_\star)}$ are chosen to densely cover the spatio-temporal domain. In PDE parameter space, this metric is defined as

$$\text{RMSE} = \sqrt{\frac{1}{N_{\boldsymbol{\theta}}}\sum_{i=1}^{N_{\boldsymbol{\theta}}}\left[\theta^{(i)} - \hat{\theta}^{(i)})\right]^2},$$

where $\boldsymbol{\theta}$ is the ground truth parameter vector and $\hat{\boldsymbol{\theta}}$ the model's prediction.

B.7.2 Mean predictive likelihood (MPL)

The *mean predictive likelihood (MPL)* is a commonly used metric for assessing how well a model predicts new data. It estimates the expected probability density that the model's predictive distribution assigns to the ground truth solution function. Formally, it is defined as (Psaros et al., 2023, Eq. 18)

$$\mathrm{MPL} = \mathbb{E}_{x,u\sim\mathcal{D}_{\text{test}}}\, \mathbb{E}_{\boldsymbol{\theta}|\mathcal{D}}\, p(u \mid \boldsymbol{x}, \boldsymbol{\theta}) \approx \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \mathcal{N}\Big(\hat{\mu}(\boldsymbol{x}_\star^{(i)}),\, \hat{\sigma}^2(\boldsymbol{x}_\star^{(i)})\Big)\,[u_i].$$

Here, for each test point i, the term $\mathcal{N}(\hat{\mu}(\boldsymbol{x}_\star^{(i)}), \hat{\sigma}^2(\boldsymbol{x}_\star^{(i)}))[u_i]$ is the value of the predictive probability density function (found using the model's predictive mean and variance) evaluated at the ground truth output value $u_i = u(\boldsymbol{x}_\star^{(i)})$. The predictive density is based on the Gaussian approximation $p\left(u_i \mid \boldsymbol{x}_\star^{(i)}, \boldsymbol{\theta}\right) \approx \mathcal{N}\left(\hat{\mu}(\boldsymbol{x}_\star^{(i)}), \hat{\sigma}^2(\boldsymbol{x}_\star^{(i)})\right)$.

A higher MPL indicates that the test data are judged more probable under the model's predictive distribution, implying stronger predictive capability.

B.7.3 Normalised Inner Product Between Variances (NIP-G)

The *Normalised Inner Product with respect to the Gold Standard (NIP-G)* quantifies the similarity between the predictive uncertainties produced by a machine learning model and those obtained from a gold standard method.

Let $\sigma_G^2(\mathbf{X}_\star)$ denote the predictive variances from the gold standard method at test locations $\mathbf{X}_\star = [\boldsymbol{x}_\star^{(1)}; \ldots; \boldsymbol{x}_\star^{(N_\star)}]^\top$, and let $\hat{\sigma}^2(\mathbf{X}_\star)$ denote the corresponding variances predicted by the model under evaluation. The NIP-G metric is defined as (Psaros et al., 2023, Eq. C.2)

$$\text{NIP-G} = \frac{\langle \sigma_G^2(\mathbf{X}_\star),\, \hat{\sigma}^2(\mathbf{X}_\star)\rangle}{\sqrt{\langle \sigma_G^2(\mathbf{X}_\star),\, \sigma_G^2(\mathbf{X}_\star)\rangle\, \langle \hat{\sigma}^2(\mathbf{X}_\star),\, \hat{\sigma}^2(\mathbf{X}_\star)\rangle}}.$$

This metric takes values in $[0, 1]$, with values close to 1 indicating strong agreement in the predicted variances, and values near 0 indicating weak correspondence.

B.7.4 Jensen–Shannon divergence (JSD)

The *Jensen–Shannon divergence (JSD)* is a symmetric and finite measure of dissimilarity between two probability distributions. It is derived from the Kullback–Leibler (KL) divergence but addresses two of its main limitations: asymmetry and potential unboundedness. By construction, the JSD treats both distributions equally and always yields a non-negative, bounded value.

Formally, given two probability densities $p(\boldsymbol{\theta})$ and $q(\boldsymbol{\theta})$, the JSD is defined as (Prince, 2023, Eq. C.35)

$$\mathbb{D}_{\text{JS}}[p(\boldsymbol{\theta}) \,\|\, q(\boldsymbol{\theta})] = \frac{1}{2}\mathbb{D}_{\text{KL}}\left[p(\boldsymbol{\theta}) \,\middle\|\, \frac{p(\boldsymbol{\theta}) + q(\boldsymbol{\theta})}{2}\right] + \frac{1}{2}\mathbb{D}_{\text{KL}}\left[q(\boldsymbol{\theta}) \,\middle\|\, \frac{p(\boldsymbol{\theta}) + q(\boldsymbol{\theta})}{2}\right].$$

Here, the mixture distribution $\frac{1}{2}\left[p(\boldsymbol{\theta}) + q(\boldsymbol{\theta})\right]$ represents the average of the two distributions. The JSD can thus be interpreted as the mean of the KL divergences from each distribution to their common average. Since both terms are weighted equally, the resulting divergence is symmetric, i.e., $\mathbb{D}_{\text{JS}}[p\|q] = \mathbb{D}_{\text{JS}}[q\|p]$.

A smaller JSD indicates greater similarity between $p(\boldsymbol{\theta})$ and $q(\boldsymbol{\theta})$, while larger values reflect increasing dissimilarity. Due to its symmetry, boundedness, and well-defined behaviour even when the supports of p and q do not perfectly overlap, the JSD is widely used for comparing predictive distributions, model uncertainty, and probabilistic forecasts.

B.7.5 Neural Network Setup

For most experiments, we employed a fully connected neural networks with depth 4 and width 128, using the tanh activation function. An exception was made for the Fisher-like equation, where a smaller architecture (depth 3, width 20) was used to help prevent overfitting.

To enforce Dirichlet boundary and initial conditions, we adopted the hard constraint approach of Sukumar and Srivastava (2022) for all PINN variants. This eliminated the need to include boundary condition terms in the likelihood (for hmcPINN, flowPINN, mfviPINN and mcPINN) or the loss function (for densPINN). Specifically, we used constrained neural networks $\tilde{u}_{\boldsymbol{\omega}}$ of the form:

$$\tilde{u}_{\boldsymbol{\omega}}(\boldsymbol{x}, t) = m(\boldsymbol{x}, t) + d(\boldsymbol{x}, t) u_{\boldsymbol{\omega}}(\boldsymbol{x}, t), \tag{B.24}$$

where $m(\boldsymbol{x}, t)$ is a function that interpolates the prescribed boundary and initial conditions, and $d(\boldsymbol{x}, t)$ is a distance function that vanishes on the boundary and remains positive elsewhere. For instance, in the `Poisson2D` example, we set $m(\boldsymbol{x}) = x_1 \sin(\pi x_2)$ and $d(\boldsymbol{x}) = x_1 x_2 (1 - x_1)(1 - x_2)$, while for the `Burgers` experiment, we used $m(x, t) = -\sin(\pi x)$ and $d(x, t) = (1 - x)(x - 1)t$.

B.7.6 Specifying the number of PDE collocation points N_f

The performance of each of the PIML methods depends on the number of PDE collocation points N_f (see Eq. 8) used to enforce the governing equation. If too few collocation points are employed, the PDE constraint is weakly enforced. Conversely, using too many collocation points can cause the posterior to collapse toward a single, biased estimate of $\boldsymbol{\theta}$. This trade-off is further influenced by the collocation noise hyperparameter σ_f.

For the VI methods (flowPINN, mfviPINN, and mcdPINN) we addressed this trade-off using the simple heuristic of setting the number of collocation points N_f equal to the number of solution observations N_u (see Eq. 7). This choice also reduced computational cost and accelerated training. To ensure adequate coverage of the spatio-temporal domain, collocation point locations were randomly resampled at each training iteration.

For hmcPINN, we manually tuned N_f (together with different values of σ_f - see Appendix B.7.7) for the best empirical performance. In contrast, densPINN employed a larger number of collocation points, as the `neuralUQ` package does not support mini-batch sampling. We found that densPINN results were relatively insensitive to the exact choice of N_f.

B.7.7 Handling of noise hyperparameters for hmcPINN

In the original B-PINN publication (Stock et al., 2023), the ground truth noise levels σ_u and σ_f used to define the likelihood function in Figure 2 (b) were assumed to be known. We adopt the same approach here, as we find that attempting to infer these noise levels adds additional complexity to the MCMC sampling problem, making it difficult to achieve good mixing.

For the observation data noise level σ_u (see Eq. 7), we set this to the ground truth value. For each experiment, we manually tune the collocation noise level σ_f (see Eq. 8) to obtain the best results against the ground truth PDE parameters $\boldsymbol{\theta}$ and solution u. If σ_f is too low, the sampler finds it hard to move through parameter space. If σ_f is too high, the PDE is not enforced sufficiently strongly. Since this tuning requires *a posteriori* information, it provides an advantage to hmcPINN relative to the other models.

C Additional results

C.1 Additional results plots

Posterior plots for the `FisherLike` and `Burgers` experiments from Section 5.2 are shown in Figure C.5, each based on a single regenerated dataset. For the `FisherLike` experiment, flowPINN and hmcPINN achieve similar accuracy in solution-space, while densPINN performs worse. In PDE parameter space, both flowPINN and hmcPINN produce approximately unbiased samples, whereas densPINN and mcdPINN exhibit noticeable bias. For the `Burgers` experiment, flowPINN achieves substantially higher accuracy in both solution and parameter space, however, all models produce parameter-space samples that are biased.

Figure C.6 presents trace plots of the three components of the parameter vector $\boldsymbol{\theta}$ for the `Poisson3D` experiment from Section 5.1, comparing the ground truth samples obtained from the nutsFEM method with those produced by the hmcPINN model. The traces for hmcPINN exhibit noticeable autocorrelation when compared to the gold standard samples.

Figure C.7 shows traceplots of the PDE parameter samples obtained using hmcPINN for the `FisherLike` and `Burgers` experiments. The sampler obtained better posterior mixing in these cases than for `Poisson3D`.

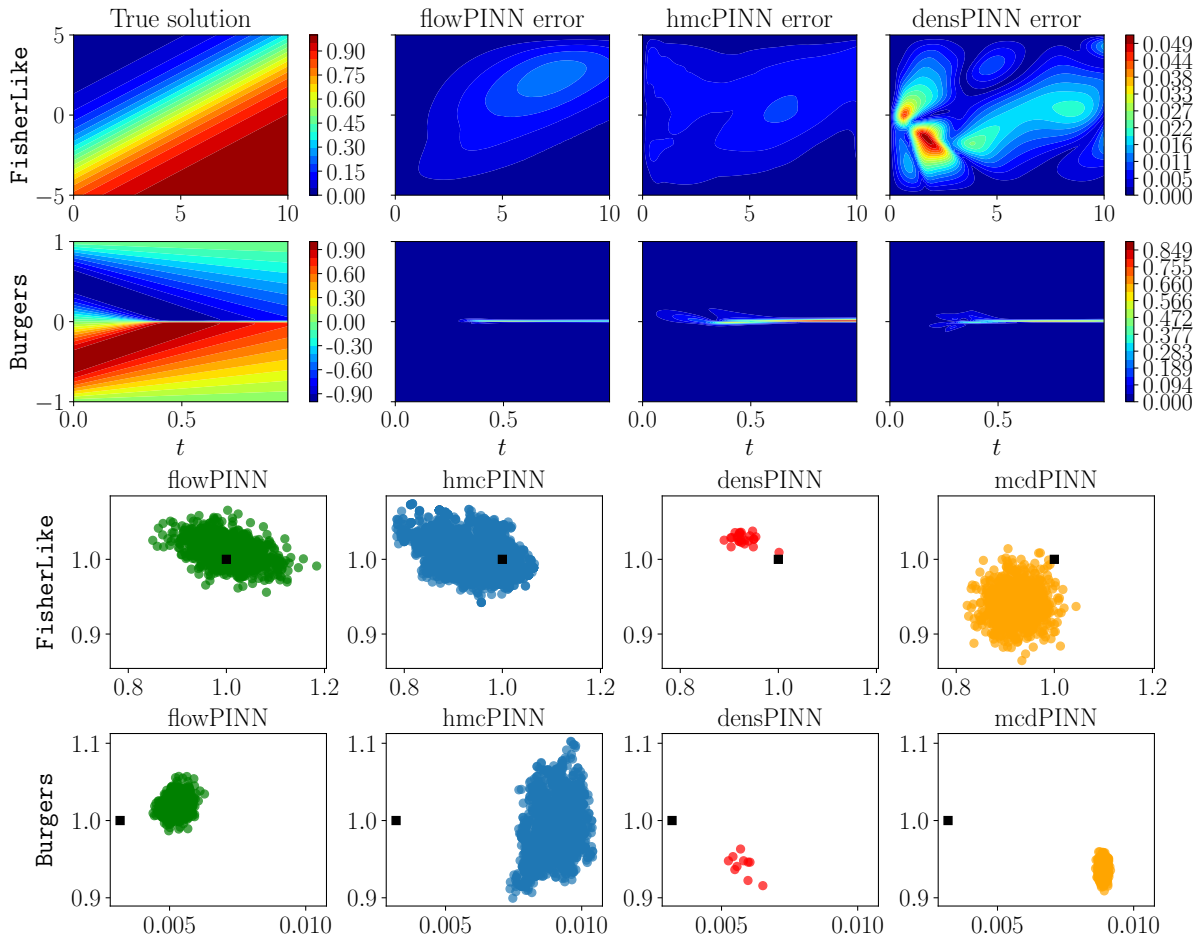


Figure C.5: `FisherLike` and `Burgers` results for flowPINN, hmcPINN, densPINN, and mcdPINN, each evaluated on a single regenerated dataset. The top two rows show density plots of the true solution and the corresponding prediction errors, while the bottom rows present scatter plots of posterior samples of the PDE parameters $\boldsymbol{\theta}$ from each model. The ground truth $\boldsymbol{\theta}$ values are indicated as black squares.

C.2 Effect of the subsample size M

We conduct a sensitivity study to assess the effect of the Stage One subsample size M (see Section 4.1) on the `Poisson2D` experiment. The results, shown in Figure C.8, present posterior samples of $\boldsymbol{\theta}$ generated by flowPINN models for different values of M. In each case, samples from the Stage One approximate posterior $q_{\boldsymbol{\phi}_1^\star}$ are shown above those from the final posterior $q_{\boldsymbol{\phi}^\star}$ obtained after Stage Three.

The choice of M has a pronounced effect on the variability of the Stage One posterior $q_{\boldsymbol{\phi}_1^\star}$. This is because smaller values of M increase the regularisation effect of the prior, leading to more diffuse samples from $q_{\boldsymbol{\phi}_1^\star}$.

For $M \leq 5$, the Stage Three posteriors remain stable. This is because the corresponding Stage One posteriors are sufficiently diffuse to cover the support of the gold standard posterior. In contrast, for larger values of M, the Stage One posteriors fail to adequately cover the gold standard posterior, which results in an overly concentrated Stage Three posterior.

We therefore select a relatively small value of M in each experiment. In this example, however, when $M = 1$, the Stage One posterior becomes excessively diffuse relative to the gold standard posterior, and more closely resembles the prior. This behaviour is reflected in the divergence of the learned σ_f for $M = 1$ (see Figure C.9 a), which essentially "switches off" the likelihood term in the ELBO involving $\boldsymbol{\theta}$. Consequently, Stage Two training has to be performed over an unnecessarily large region of the PDE parameter space $\boldsymbol{\Theta}$. As a result, in this work we adopt the strategy of setting M to the smallest value that yields a stable estimate of σ_f; in this case, $M = 2$.

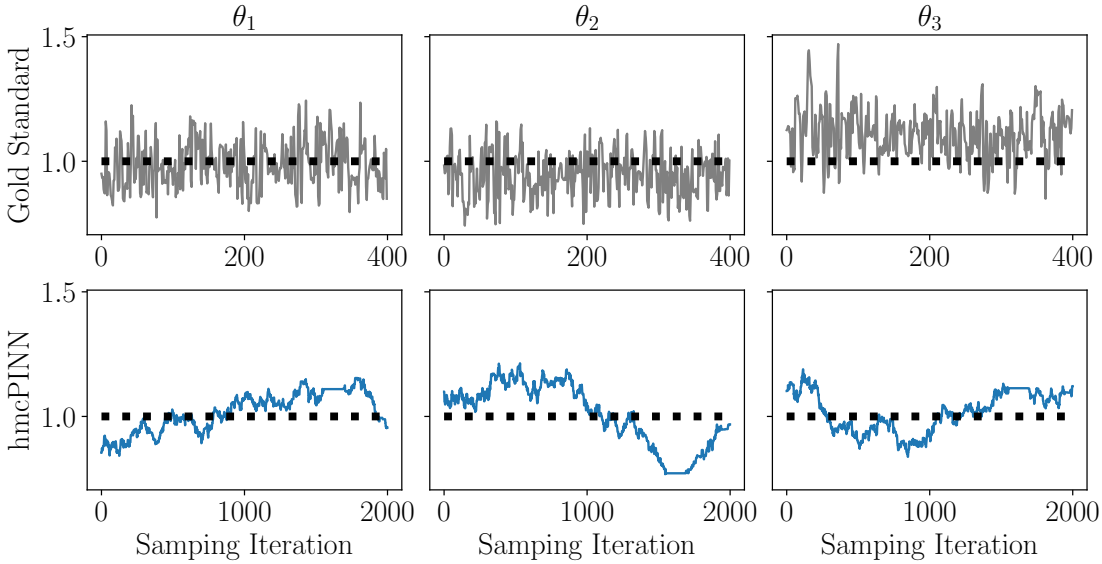


Figure C.6: Trace plots of the three components of the parameter vector $\boldsymbol{\theta}$ for the `Poisson3D` experiment, comparing the gold standard samples obtained from the nutsFEM method with those produced by hmcPINN. The dotted lines represents the ground truth parameter values. The hmcPINN samples exhibit significantly more autocorrelation than samples from the gold standard posterior. Further discussion is provided in Appendix C.1.

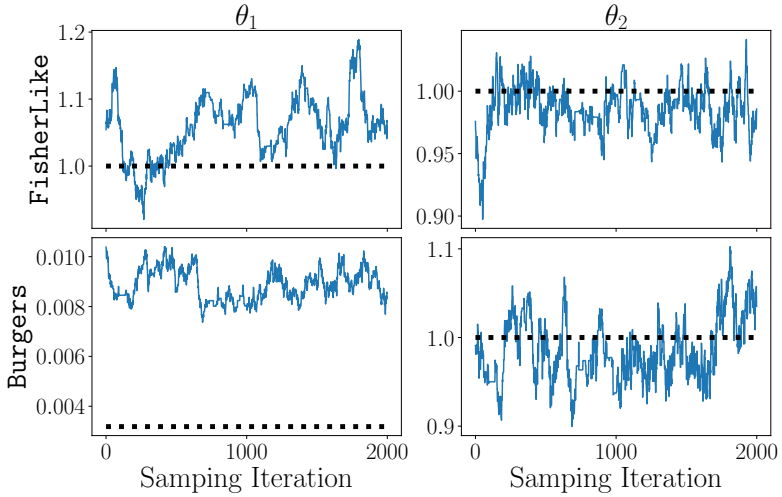


Figure C.7: Traceplots of the PDE parameter samples obtained using hmcPINN for the `FisherLike` and `Burgers` PDEs. Further discussion is provided in Appendix C.1.

Figure C.10 compares posterior samples for `Poisson2D` from the gold standard nutsFEM method with those obtained using two flowPINN configurations: a model trained in three stages with $M = 2$, and model trained in a single stage one the full data ELBO (i.e. $M = N_u = 25$). The full data flowPINN posterior is visually slightly under-dispersed, and incurs a higher JSD value. In addition, using $M = 2$ provides an approximate 40% increase in training speed in this setting, due to more efficient Stage One training enabled by the dataset subsampling.

C.3 Effect of the number of flows K

Figure C.11 presents posterior samples from the `Poisson2D` experiment described in Section 5.1, obtained using a flowPINN with varying depths K of the normalising flow component (see Eq. 3). A subsample size of $M = 2$ was used in each case. The results suggest that the inferred posterior is relatively insensitive to the choice of $K > 1$ in this setting. However, this robustness may not persist for problems where the posterior distribution exhibits more

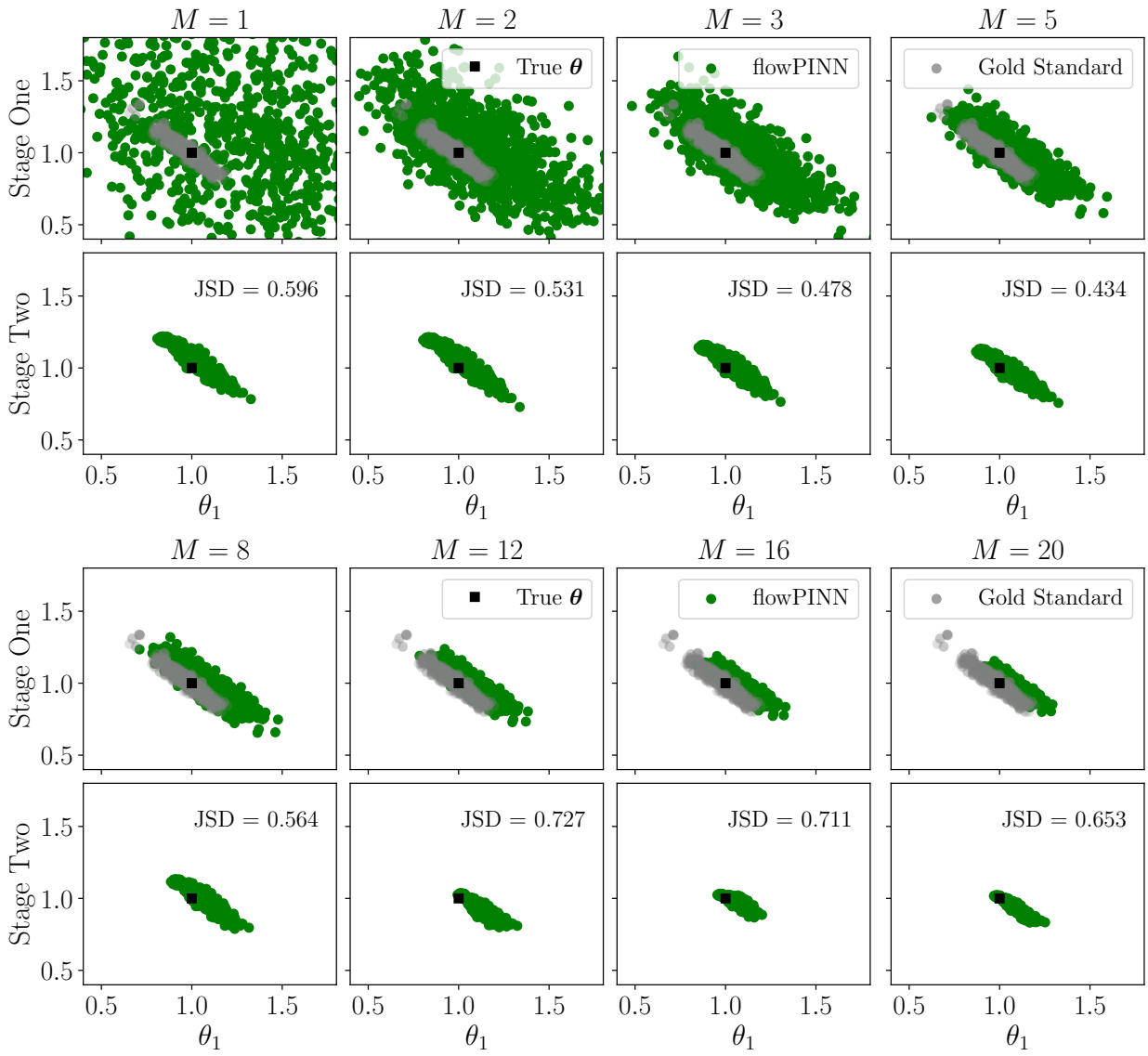


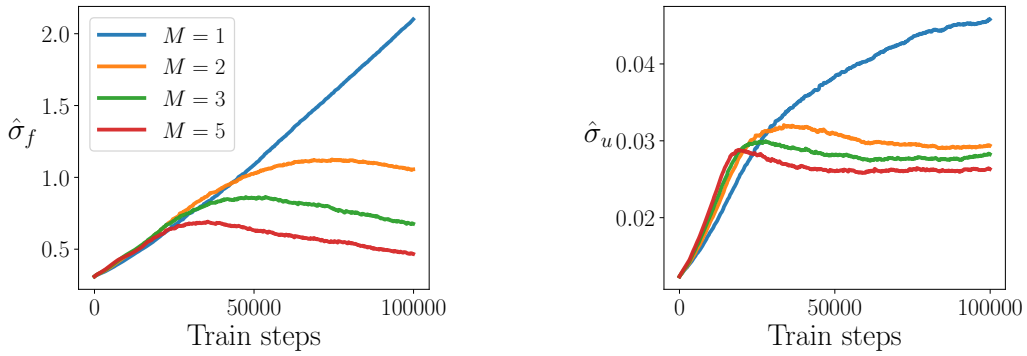
Figure C.8: Scatter plots of flowPINN $\boldsymbol{\theta}$ samples for the `Poisson2D` experiment with different subsample sizes M (see Eq. 12) with $N_u = 25$. For each M, Stage One samples (obtained using $q_{\phi_1^\star}$) are shown above Stage Three samples (obtained using the final trained flow $q_{\phi^\star}$). The choice of M strongly influences the variability of the Stage One posterior, whereas the Stage Three posteriors remain relatively stable for each value of $M \leq 5$. For larger values of M, $q_{\phi_1^\star}$ does not cover the gold standard posterior, and the final Stage Three posterior becomes under dispersed. Additional discussion is provided in Appendix C.2.

complex or highly non-Gaussian structure.

C.4 Inference with one observation point

As described in Section 4.1, Stage One of our training algorithm evaluates the ELBO (Eq. 12) using subsamples of size $M < N_u$. However, when only a single observation is available ($N_u = 1$), subsampling cannot be applied. To assess flowPINN performance in this extreme setting, we repeat the `Poisson2D` experiment using a dataset D_u (see Eq. 7) consisting of just one observation. For comparison, we also compute results using nutsFEM and hmcPINN under the same setup.

Posterior samples obtained from the three approaches are shown in Figure C.12. The reference nutsFEM posterior (left panel) is broad and exhibits a nonlinear coupling between θ_1 and θ_2. This structure is captured



(a) (b)

Figure C.9: The left (right) panel shows the training curves from the `Poisson2D` experiment, for the learned values of σ_f (σ_u) for flowPINNs with different choices of M in Stage One Training (see Eq. 12). The limited information in the likelihood when $M = 1$ is evident from the divergence of the estimate $\hat{\sigma}_f$ (see Eq. 8). In practice, we select the smallest value of M that yields a stable $\hat{\sigma}_f$ during Stage One training; in this example, that value is $M = 2$. Further discussion can be found in Appendix C.2.

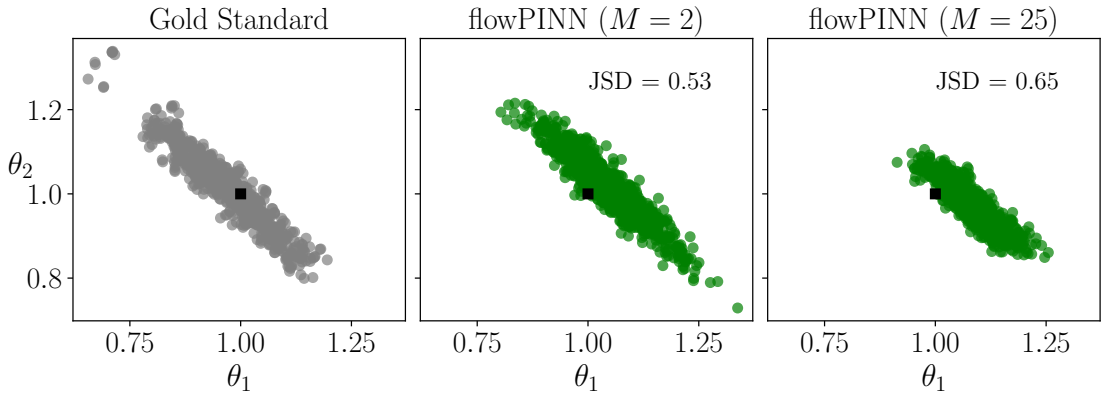


Figure C.10: Comparison of gold standard and flowPINN posterior samples from the `Poisson2D` experiment. The flowPINN trained in three stages with $M = 2$ better captures the dispersion of the gold standard posterior compared to a flowPINN trained in a single stage using the full dataset (i.e. $M = N_u = 25$). Further discussion is provided in Appendix C.2.

much more effectively by the flowPINN posterior (centre panel) than by the hmcPINN posterior (right panel). Moreover, obtaining reasonable results from hmcPINN required extensive manual tuning of the collocation noise hyperparameter σ_f (see Appendix B.7.7).

C.5 Alternative observation noise models

Our assumption in Eq. (7) that the observation noise was iid Gaussian can easily be relaxed to account for other noise models. To illustrate this, we show an example of heavy-tailed observation noise in Appendix C.5.1 below, and spatially varying noise in Appendix C.5.2.

C.5.1 Heavy tailed observation noise

We re-run the `Poisson2D` experiment using Cauchy-distributed observation noise (Johnson et al., 1995, Ch. 16). The Cauchy distribution, which has heavier tails than the Gaussian, is often used in robust regression. Figure C.13 shows the posterior results of each model. The flowPINN samples exhibit a slight bias compared to the gold standard nutsFEM samples but are otherwise well aligned. The posterior predictive distributions produced by

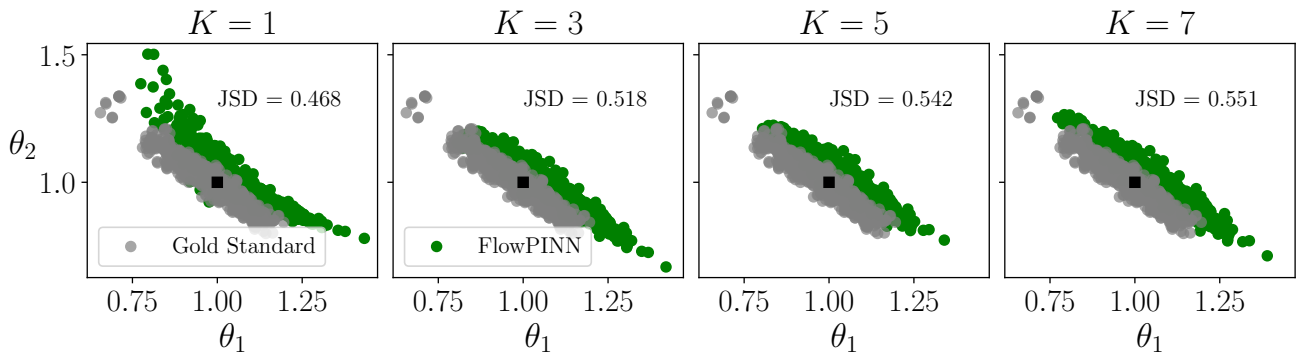


Figure C.11: Scatter plots of posterior samples of $\boldsymbol{\theta}$ for the `Poisson2D` experiment, obtained using flowPINNs with $K = 1, 3, 5, 7$ normalising flow transformations, respectively (see Eq. 3). Samples obtained using nutsFEM are considered the gold standard. The flowPINN results are relatively insensitive to K. Further discussion is provided in Appendix C.3.

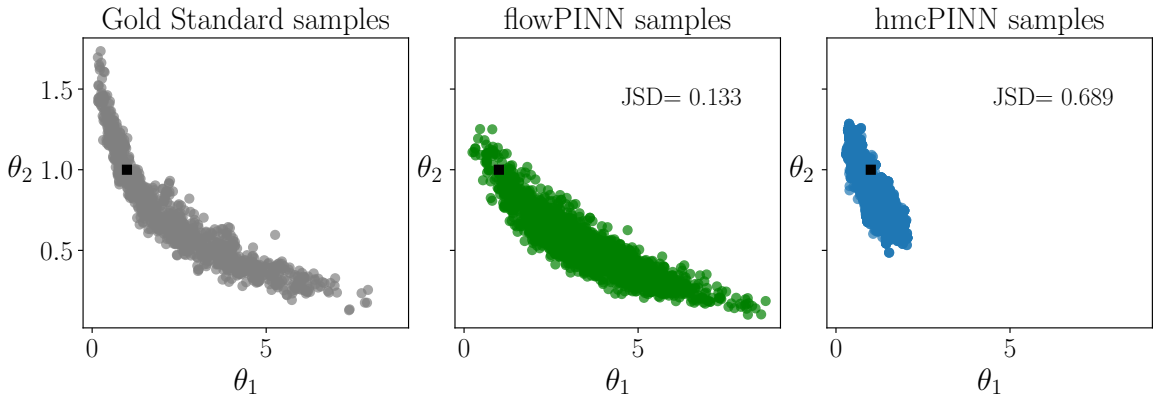


Figure C.12: Posterior samples for the `Poisson2D` experiment given $N_u = 1$ observation point (see Eq. 7), using nutsFEM (considered the gold standard), flowPINN and hmcPINN, respectively. The true parameter value is represented by a black square in each plot. The flowPINN samples provide a better approximation to the nutsFEM samples, when compared to the hmcPINN samples. Further discussion is provided in Appendix C.4.

the two models are nearly identical.

C.5.2 Spatially varying observation noise

We repeat the diffusion–reaction experiment from Appendix A using $N_u = 12$ noise-corrupted observations. In contrast to Eq. (7), here we allow standard deviation of the observation noise $\sigma_u(x)$ to depend on space. This noise field is defined as a quadratic function of x that peaks at $x = 0.5$ (see the black dotted line in the right panel of Figure C.14). We parametrise $\sigma_u(x)$ with a separate neural network, whose parameters are optimised jointly with the flow parameters during Stage Three training (see Eq. 14).

The posterior results are shown in Figure C.14. In the centre panel, flowPINN samples closely recover the true solution. The parameter samples in the left panel exhibit a well-defined linear relationship, the variance of which narrows as θ_1 increases. The noise samples in the right panel provide a qualitatively good approximation to the quadratic distribution of the true noise level.

D Future work

Extension to ill-posed problems. The posterior factorisation from Eq. (9) is not valid for ill-posed inverse problems, in which case the solution function u is not uniquely determined even when the parameters $\boldsymbol{\theta}$ are known. Such scenarios arise, for example, when boundary conditions are only partially specified, or when the

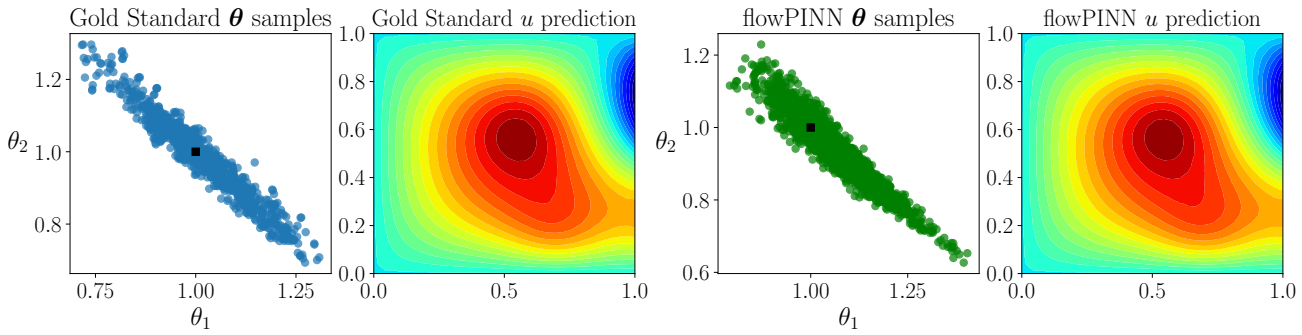


Figure C.13: Results plots for the `Poisson2D` model under heavy-tailed Cauchy observation noise for flowPINN and the gold standard nutsFEM reference. The true parameter value is represented by a black square in each plot. The flowPINN samples exhibit a slight bias compared to the gold standard nutsFEM samples but are otherwise well aligned. The posterior predictive distributions produced by the two models are nearly identical. Further discussion is provided in Appendix C.5.1.

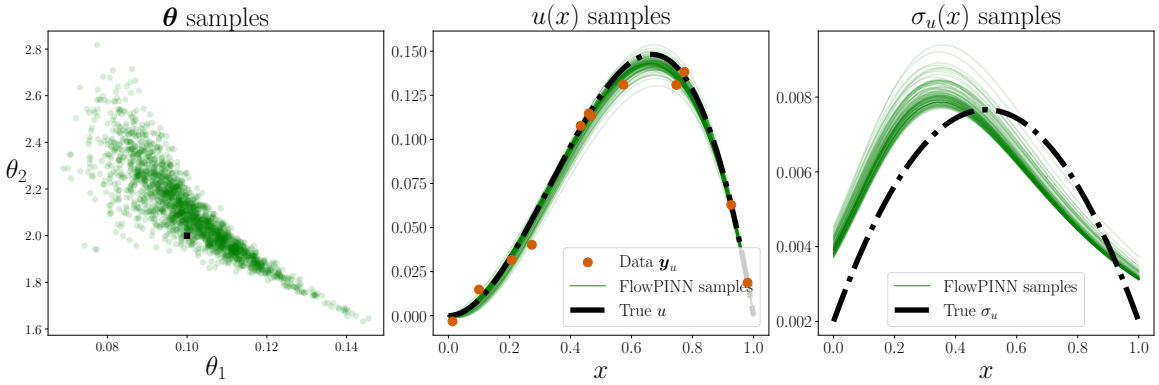


Figure C.14: FlowPINN results for the 1D reaction–diffusion PDE in Eq. (A.15) with a spatially varying observation noise level $\sigma_u(x)$ (see Eq. 7). The left panel shows posterior samples of the PDE parameters $\boldsymbol{\theta}$, the centre panel shows samples of the solution function, and the right panel shows samples of the noise levels. Ground truth values are indicated by black lines or squares. Further discussion is provided in Appendix C.5.2.

governing physical laws are incomplete or uncertain. In future work, we aim to extend our flowPINN framework to handle these cases. One potential direction is to treat the PINN component in Eq. (10) using a variational Bayesian approach, which may enable the model to capture and propagate the additional sources of uncertainty inherent in ill-posed settings.

Extension to parameter fields. Throughout this work we have assumed that the PDE parameters $\boldsymbol{\theta}$ are constant across the spatio-temporal domain - see (Tarantola, 2005, Ch. 1) and Xun et al. (2013) for further details. This assumption is standard in the majority of PDE inverse problems, including in disciplines such as biophysiology (Gao et al., 2015; Buoso et al., 2021; Maso Talou et al., 2021), epidemiology (Smirnova and Chowell, 2017; Osi and Ghaffarzadegan, 2024; Hong et al., 2024), solid mechanics (Pacheo et al., 2016; Hartmann and Gilbert, 2018; Rappel et al., 2020), systems biology (Gábor and Banga, 2015; Abdulla and Poteau, 2018; Duk et al., 2021) and fluid dynamics (Fan et al., 2010; New et al., 2024; Du et al., 2025).

Nevertheless, there do exist settings where the parameters vary in space and/or time, requiring the inference of an entire parameter field (Girolami et al., 2021). Our present formulation of $u_{\boldsymbol{\omega}}$ in Eq. (10), which employs a standard fully-connected PINN architecture, is not directly suited to such cases. This is because the model does not account for local variations in $\boldsymbol{\theta}$ when it makes a predictions at a given input location, and hence is not appropriate for spatially varying parameters. Extending flowPINNs to these scenarios would therefore require a different choice of surrogate model in Eq. (10). This could be achieved by replacing the fully-connected architecture with an approach that naturally is designed for field data, such as graph neural networks (GNNs) or

neural operators. These approaches could in principle allow flowPINNs to capture heterogeneous parameter fields, while retaining the probabilistic framework for uncertainty quantification.